Doxygen Book

Generated by Doxygen 1.8.17

1 Todo List 1

2 Bug List 5

3 Namespace Index 17

3.1 Namespace List . . . 17

4 Hierarchical Index 19

4.1 Class Hierarchy . . . 19

5 Class Index 27

5.1 Class List . . . 27

6 File Index 39

6.1 File List . . . 39

7 Namespace Documentation 49

7.1 nntrainer Namespace Reference . . . 49

7.1.1 Detailed Description . . . 58

7.1.2 Typedef Documentation . . . 58

7.1.2.1 VarGradSpec . . . 58

7.1.2.2 WeightSpec . . . 59

7.1.3 Enumeration Type Documentation . . . 59

7.1.3.1 ActivationType . . . 59

7.1.3.2 AttentionParams [1/2] . . . 59

7.1.3.3 AttentionParams [2/2] . . . 60

7.1.3.4 CachePolicy . . . 60

7.1.3.5 ExecutionMode . . . 60

7.1.3.6 INOUT_INDEX . . . 62

7.1.3.7 InPlace . . . 62

7.1.3.8 MoLAttentionParams . . . 62

7.1.3.9 PrintOption . . . 63

7.1.3.10 TensorLifespan . . . 63

7.1.3.11 WeightRegularizer . . . 64

7.1.4 Function Documentation . . . 64

7.1.4.1 add_default_object() . . . 64

7.1.4.2 AppContext::registerFactory< ml::train::LearningRateScheduler >() . . . 65

7.1.4.3 AppContext::registerFactory< nntrainer::Layer >() . . . 65

7.1.4.4 AppContext::registerFactory< nntrainer::Optimizer >() . . . 65

7.1.4.5 buildTrasposeString() . . . 65

7.1.4.6 computeSpace() . . . 66

7.1.4.7 createLayerNode() [1/3] . . . 66

7.1.4.8 createLayerNode() [2/3] . . . 67

7.1.4.9 createLayerNode() [3/3] . . . 68

7.1.4.10 fini_global_context_nntrainer() . . . 68
7.1.4.11 grucell_calcGradient() . . . 68
7.1.4.12 grucell_forwarding() . . . 69
7.1.4.13 iterate_prop() [1/3] . . . 69
7.1.4.14 iterate_prop() [2/3] . . . 70
7.1.4.15 iterate_prop() [3/3] . . . 70
7.1.4.16 loadProperties() . . . 71
7.1.4.17 overlap() . . . 72
7.1.4.18 parse_properties() . . . 72
7.1.4.19 propagateTimestep() . . . 73
7.1.4.20 setInplaceSharedMemoryConfigByLayer() . . . 74
7.1.4.21 suffixSpec() . . . 74
7.1.4.22 validateIntervalOverlap() [1/3] . . . 75
7.1.4.23 validateIntervalOverlap() [2/3] . . . 75
7.1.4.24 validateIntervalOverlap() [3/3] . . . 76
7.1.5 Variable Documentation . . . 76
7.1.5.1 createfunc . . . 76
7.1.5.2 rng [1/2] . . . 77
7.1.5.3 rng [2/2] . . . 77
7.2 nntrainer::exception Namespace Reference . . . 77
7.2.1 Detailed Description . . . 77

8 Class Documentation **79**

8.1 tflite::AbsOptionsBuilder Struct Reference . . . 79
8.1.1 Detailed Description . . . 79
8.2 nntrainer::props::Activation Class Reference . . . 80
8.2.1 Detailed Description . . . 81
8.3 nntrainer::ActivationRealizer Class Reference . . . 81
8.3.1 Detailed Description . . . 82
8.3.2 Constructor & Destructor Documentation . . . 82
8.3.2.1 ~ActivationRealizer() . . . 82
8.3.3 Member Function Documentation . . . 82
8.3.3.1 realize() . . . 82
8.4 nntrainer::props::ActivationTypeInfo Struct Reference . . . 83
8.4.1 Detailed Description . . . 83
8.4.2 Member Data Documentation . . . 83
8.4.2.1 EnumList . . . 83
8.4.2.2 EnumStr . . . 83
8.5 tflite::AddNOptionsBuilder Struct Reference . . . 84
8.5.1 Detailed Description . . . 84
8.6 tflite::AddOptionsBuilder Struct Reference . . . 84
8.6.1 Detailed Description . . . 85

8.7 nntrainer::AppContext Class Reference . . . 85
8.7.1 Detailed Description . . . 86
8.7.2 Member Typedef Documentation . . . 86
8.7.2.1 IndexType . . . 86
8.7.2.2 IntIndexType . . . 87
8.7.3 Member Function Documentation . . . 87
8.7.3.1 createObject() [1/2] . . . 87
8.7.3.2 createObject() [2/2] . . . 87
8.7.3.3 getProperties() . . . 88
8.7.3.4 getWorkingPath() . . . 88
8.7.3.5 Global() . . . 89
8.7.3.6 hasWorkingDirectory() . . . 89
8.7.3.7 registerFactory() [1/2] . . . 89
8.7.3.8 registerFactory() [2/2] . . . 90
8.7.3.9 registerLayer() . . . 91
8.7.3.10 registerOptimizer() . . . 92
8.7.3.11 registerPluggableFromDirectory() . . . 92
8.7.3.12 setWorkingDirectory() . . . 92
8.7.3.13 unknownFactory() . . . 94
8.7.3.14 unsetWorkingDirectory() . . . 94
8.8 tflite::ArgMaxOptionsBuilder Struct Reference . . . 95
8.8.1 Detailed Description . . . 95
8.9 tflite::ArgMinOptionsBuilder Struct Reference . . . 95
8.9.1 Detailed Description . . . 96
8.10 nntrainer::props::AsSequence Class Reference . . . 96
8.10.1 Detailed Description . . . 97
8.11 nntrainer::props::AverageAttentionWeight Class Reference . . . 97
8.11.1 Detailed Description . . . 98
8.11.2 Member Typedef Documentation . . . 98
8.11.2.1 prop_tag . . . 98
8.11.3 Member Data Documentation . . . 98
8.11.3.1 key . . . 98
8.12 nntrainer::props::Axis Class Reference . . . 99
8.12.1 Detailed Description . . . 100
8.12.2 Member Typedef Documentation . . . 100
8.12.2.1 prop_tag . . . 100
8.12.3 Member Function Documentation . . . 100
8.12.3.1 isValid() . . . 100
8.12.4 Member Data Documentation . . . 101
8.12.4.1 key . . . 101
8.13 nntrainer::Backend Class Reference . . . 101
8.13.1 Detailed Description . . . 102

8.13.2 Member Enumeration Documentation . . . 102
8.13.2.1 BackendType . . . 102
8.13.3 Constructor & Destructor Documentation . . . 102
8.13.3.1 Backend() . . . 102
8.13.4 Member Function Documentation . . . 103
8.13.4.1 setNumThreads() . . . 103
8.14 nntrainer::BasicPlanner Class Reference . . . 103
8.14.1 Detailed Description . . . 104
8.14.2 Constructor & Destructor Documentation . . . 104
8.14.2.1 ∼BasicPlanner() . . . 104
8.14.3 Member Function Documentation . . . 105
8.14.3.1 getType() . . . 105
8.14.3.2 planLayout() . . . 105
8.15 nntrainer::props::BasicRegularizer Class Reference . . . 106
8.15.1 Detailed Description . . . 107
8.15.2 Member Function Documentation . . . 107
8.15.2.1 isValid() . . . 107
8.16 nntrainer::props::BasicRegularizerConstant Class Reference . . . 107
8.16.1 Detailed Description . . . 108
8.16.2 Member Typedef Documentation . . . 109
8.16.2.1 prop_tag . . . 109
8.16.3 Constructor & Destructor Documentation . . . 109
8.16.3.1 BasicRegularizerConstant() . . . 109
8.16.4 Member Function Documentation . . . 109
8.16.4.1 isValid() . . . 109
8.16.5 Member Data Documentation . . . 110
8.16.5.1 key . . . 110
8.17 tflite::BatchMatMulOptionsBuilder Struct Reference . . . 110
8.17.1 Detailed Description . . . 110
8.18 tflite::BatchToSpaceNDOptionsBuilder Struct Reference . . . 111
8.18.1 Detailed Description . . . 111
8.19 nntrainer::props::BiasDecay Class Reference . . . 111
8.19.1 Detailed Description . . . 112
8.19.2 Constructor & Destructor Documentation . . . 112
8.19.2.1 BiasDecay() . . . 112
8.19.3 Member Data Documentation . . . 113
8.19.3.1 key . . . 113
8.20 nntrainer::props::BiasInitializer Class Reference . . . 113
8.20.1 Detailed Description . . . 114
8.21 nntrainer::props::Bidirectional Class Reference . . . 114
8.21.1 Detailed Description . . . 115
8.21.2 Constructor & Destructor Documentation . . . 115

8.21.2.1 Bidirectional() . . . 115
8.22 tflite::BidirectionalSequenceLSTMOptionsBuilder Struct Reference . . . 116
8.22.1 Detailed Description . . . 116
8.23 tflite::BidirectionalSequenceRNNOptionsBuilder Struct Reference . . . 116
8.23.1 Detailed Description . . . 117
8.24 nntrainer::props::BNPARAMS_BETA_INIT Class Reference . . . 117
8.24.1 Detailed Description . . . 118
8.25 nntrainer::props::BNPARAMS_GAMMA_INIT Class Reference . . . 118
8.25.1 Detailed Description . . . 119
8.26 nntrainer::props::BNPARAMS_MU_INIT Class Reference . . . 120
8.26.1 Detailed Description . . . 121
8.27 nntrainer::props::BNPARAMS_VAR_INIT Class Reference . . . 121
8.27.1 Detailed Description . . . 122
8.28 nntrainer::BnRealizer Class Reference . . . 122
8.28.1 Detailed Description . . . 123
8.28.2 Constructor & Destructor Documentation . . . 123
8.28.2.1 BnRealizer() . . . 123
8.28.2.2 ∼BnRealizer() . . . 124
8.28.3 Member Function Documentation . . . 124
8.28.3.1 realize() . . . 124
8.29 nntrainer::Tensor::BroadcastInfo Struct Reference . . . 125
8.29.1 Detailed Description . . . 125
8.29.2 Constructor & Destructor Documentation . . . 125
8.29.2.1 BroadcastInfo() . . . 125
8.29.3 Member Data Documentation . . . 125
8.29.3.1 buffer_axis . . . 125
8.29.3.2 buffer_size . . . 126
8.29.3.3 strides . . . 126
8.30 tflite::BufferBuilder Struct Reference . . . 126
8.30.1 Detailed Description . . . 126
8.31 tflite::BuiltinOptionsTraits< T > Struct Template Reference . . . 127
8.31.1 Detailed Description . . . 127
8.32 tflite::BuiltinOptionsTraits< tflite::AbsOptions > Struct Reference . . . 127
8.32.1 Detailed Description . . . 127
8.33 tflite::BuiltinOptionsTraits< tflite::AddNOptions > Struct Reference . . . 127
8.33.1 Detailed Description . . . 127
8.34 tflite::BuiltinOptionsTraits< tflite::AddOptions > Struct Reference . . . 128
8.34.1 Detailed Description . . . 128
8.35 tflite::BuiltinOptionsTraits< tflite::ArgMaxOptions > Struct Reference . . . 128
8.35.1 Detailed Description . . . 128
8.36 tflite::BuiltinOptionsTraits< tflite::ArgMinOptions > Struct Reference . . . 128
8.36.1 Detailed Description . . . 128

8.37 tflite::BuiltinOptionsTraits< tflite::BatchMatMulOptions > Struct Reference 129
8.37.1 Detailed Description 129
8.38 tflite::BuiltinOptionsTraits< tflite::BatchToSpaceNDOptions > Struct Reference 129
8.38.1 Detailed Description 129
8.39 tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceLSTMOptions > Struct Reference 129
8.39.1 Detailed Description 129
8.40 tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceRNNOptions > Struct Reference 130
8.40.1 Detailed Description 130
8.41 tflite::BuiltinOptionsTraits< tflite::CallOptions > Struct Reference 130
8.41.1 Detailed Description 130
8.42 tflite::BuiltinOptionsTraits< tflite::CastOptions > Struct Reference 130
8.42.1 Detailed Description 130
8.43 tflite::BuiltinOptionsTraits< tflite::ConcatEmbeddingsOptions > Struct Reference 131
8.43.1 Detailed Description 131
8.44 tflite::BuiltinOptionsTraits< tflite::ConcatenationOptions > Struct Reference 131
8.44.1 Detailed Description 131
8.45 tflite::BuiltinOptionsTraits< tflite::Conv2DOptions > Struct Reference 131
8.45.1 Detailed Description 131
8.46 tflite::BuiltinOptionsTraits< tflite::CosOptions > Struct Reference 132
8.46.1 Detailed Description 132
8.47 tflite::BuiltinOptionsTraits< tflite::CumsumOptions > Struct Reference 132
8.47.1 Detailed Description 132
8.48 tflite::BuiltinOptionsTraits< tflite::DensifyOptions > Struct Reference 132
8.48.1 Detailed Description 132
8.49 tflite::BuiltinOptionsTraits< tflite::DepthToSpaceOptions > Struct Reference 133
8.49.1 Detailed Description 133
8.50 tflite::BuiltinOptionsTraits< tflite::DepthwiseConv2DOptions > Struct Reference 133
8.50.1 Detailed Description 133
8.51 tflite::BuiltinOptionsTraits< tflite::DequantizeOptions > Struct Reference 133
8.51.1 Detailed Description 133
8.52 tflite::BuiltinOptionsTraits< tflite::DivOptions > Struct Reference 134
8.52.1 Detailed Description 134
8.53 tflite::BuiltinOptionsTraits< tflite::EmbeddingLookupSparseOptions > Struct Reference 134
8.53.1 Detailed Description 134
8.54 tflite::BuiltinOptionsTraits< tflite::EqualOptions > Struct Reference 134
8.54.1 Detailed Description 134
8.55 tflite::BuiltinOptionsTraits< tflite::ExpandDimsOptions > Struct Reference 135
8.55.1 Detailed Description 135
8.56 tflite::BuiltinOptionsTraits< tflite::ExpOptions > Struct Reference 135
8.56.1 Detailed Description 135
8.57 tflite::BuiltinOptionsTraits< tflite::FakeQuantOptions > Struct Reference 135
8.57.1 Detailed Description 135

8.58 tflite::BuiltinOptionsTraits< tflite::FillOptions > Struct Reference . . . 136
8.58.1 Detailed Description . . . 136
8.59 tflite::BuiltinOptionsTraits< tflite::FloorDivOptions > Struct Reference . . . 136
8.59.1 Detailed Description . . . 136
8.60 tflite::BuiltinOptionsTraits< tflite::FloorModOptions > Struct Reference . . . 136
8.60.1 Detailed Description . . . 136
8.61 tflite::BuiltinOptionsTraits< tflite::FullyConnectedOptions > Struct Reference . . . 137
8.61.1 Detailed Description . . . 137
8.62 tflite::BuiltinOptionsTraits< tflite::GatherNdOptions > Struct Reference . . . 137
8.62.1 Detailed Description . . . 137
8.63 tflite::BuiltinOptionsTraits< tflite::GatherOptions > Struct Reference . . . 137
8.63.1 Detailed Description . . . 137
8.64 tflite::BuiltinOptionsTraits< tflite::GreaterEqualOptions > Struct Reference . . . 138
8.64.1 Detailed Description . . . 138
8.65 tflite::BuiltinOptionsTraits< tflite::GreaterOptions > Struct Reference . . . 138
8.65.1 Detailed Description . . . 138
8.66 tflite::BuiltinOptionsTraits< tflite::HardSwishOptions > Struct Reference . . . 138
8.66.1 Detailed Description . . . 138
8.67 tflite::BuiltinOptionsTraits< tflite::IfOptions > Struct Reference . . . 139
8.67.1 Detailed Description . . . 139
8.68 tflite::BuiltinOptionsTraits< tflite::L2NormOptions > Struct Reference . . . 139
8.68.1 Detailed Description . . . 139
8.69 tflite::BuiltinOptionsTraits< tflite::LeakyReluOptions > Struct Reference . . . 139
8.69.1 Detailed Description . . . 139
8.70 tflite::BuiltinOptionsTraits< tflite::LessEqualOptions > Struct Reference . . . 140
8.70.1 Detailed Description . . . 140
8.71 tflite::BuiltinOptionsTraits< tflite::LessOptions > Struct Reference . . . 140
8.71.1 Detailed Description . . . 140
8.72 tflite::BuiltinOptionsTraits< tflite::LocalResponseNormalizationOptions > Struct Reference . . . 140
8.72.1 Detailed Description . . . 140
8.73 tflite::BuiltinOptionsTraits< tflite::LogicalAndOptions > Struct Reference . . . 141
8.73.1 Detailed Description . . . 141
8.74 tflite::BuiltinOptionsTraits< tflite::LogicalNotOptions > Struct Reference . . . 141
8.74.1 Detailed Description . . . 141
8.75 tflite::BuiltinOptionsTraits< tflite::LogicalOrOptions > Struct Reference . . . 141
8.75.1 Detailed Description . . . 141
8.76 tflite::BuiltinOptionsTraits< tflite::LogSoftmaxOptions > Struct Reference . . . 142
8.76.1 Detailed Description . . . 142
8.77 tflite::BuiltinOptionsTraits< tflite::LSHProjectionOptions > Struct Reference . . . 142
8.77.1 Detailed Description . . . 142
8.78 tflite::BuiltinOptionsTraits< tflite::LSTMOptions > Struct Reference . . . 142
8.78.1 Detailed Description . . . 142

8.79 tflite::BuiltinOptionsTraits< tflite::MatrixDiagOptions > Struct Reference 143
8.79.1 Detailed Description 143
8.80 tflite::BuiltinOptionsTraits< tflite::MatrixSetDiagOptions > Struct Reference 143
8.80.1 Detailed Description 143
8.81 tflite::BuiltinOptionsTraits< tflite::MaximumMinimumOptions > Struct Reference 143
8.81.1 Detailed Description 143
8.82 tflite::BuiltinOptionsTraits< tflite::MirrorPadOptions > Struct Reference 144
8.82.1 Detailed Description 144
8.83 tflite::BuiltinOptionsTraits< tflite::MulOptions > Struct Reference 144
8.83.1 Detailed Description 144
8.84 tflite::BuiltinOptionsTraits< tflite::NegOptions > Struct Reference 144
8.84.1 Detailed Description 144
8.85 tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV4Options > Struct Reference 145
8.85.1 Detailed Description 145
8.86 tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV5Options > Struct Reference 145
8.86.1 Detailed Description 145
8.87 tflite::BuiltinOptionsTraits< tflite::NotEqualOptions > Struct Reference 145
8.87.1 Detailed Description 145
8.88 tflite::BuiltinOptionsTraits< tflite::OneHotOptions > Struct Reference 146
8.88.1 Detailed Description 146
8.89 tflite::BuiltinOptionsTraits< tflite::PackOptions > Struct Reference 146
8.89.1 Detailed Description 146
8.90 tflite::BuiltinOptionsTraits< tflite::PadOptions > Struct Reference 146
8.90.1 Detailed Description 146
8.91 tflite::BuiltinOptionsTraits< tflite::PadV2Options > Struct Reference 147
8.91.1 Detailed Description 147
8.92 tflite::BuiltinOptionsTraits< tflite::Pool2DOptions > Struct Reference 147
8.92.1 Detailed Description 147
8.93 tflite::BuiltinOptionsTraits< tflite::PowOptions > Struct Reference 147
8.93.1 Detailed Description 147
8.94 tflite::BuiltinOptionsTraits< tflite::QuantizeOptions > Struct Reference 148
8.94.1 Detailed Description 148
8.95 tflite::BuiltinOptionsTraits< tflite::RangeOptions > Struct Reference 148
8.95.1 Detailed Description 148
8.96 tflite::BuiltinOptionsTraits< tflite::RankOptions > Struct Reference 148
8.96.1 Detailed Description 148
8.97 tflite::BuiltinOptionsTraits< tflite::ReducerOptions > Struct Reference 149
8.97.1 Detailed Description 149
8.98 tflite::BuiltinOptionsTraits< tflite::ReshapeOptions > Struct Reference 149
8.98.1 Detailed Description 149
8.99 tflite::BuiltinOptionsTraits< tflite::ResizeBilinearOptions > Struct Reference 149
8.99.1 Detailed Description 149

8.100 tflite::BuiltinOptionsTraits< tflite::ResizeNearestNeighborOptions > Struct Reference 150
8.100.1 Detailed Description 150
8.101 tflite::BuiltinOptionsTraits< tflite::ReverseSequenceOptions > Struct Reference 150
8.101.1 Detailed Description 150
8.102 tflite::BuiltinOptionsTraits< tflite::ReverseV2Options > Struct Reference 150
8.102.1 Detailed Description 150
8.103 tflite::BuiltinOptionsTraits< tflite::RNNOptions > Struct Reference 151
8.103.1 Detailed Description 151
8.104 tflite::BuiltinOptionsTraits< tflite::ScatterNdOptions > Struct Reference 151
8.104.1 Detailed Description 151
8.105 tflite::BuiltinOptionsTraits< tflite::SegmentSumOptions > Struct Reference 151
8.105.1 Detailed Description 151
8.106 tflite::BuiltinOptionsTraits< tflite::SelectOptions > Struct Reference 152
8.106.1 Detailed Description 152
8.107 tflite::BuiltinOptionsTraits< tflite::SelectV2Options > Struct Reference 152
8.107.1 Detailed Description 152
8.108 tflite::BuiltinOptionsTraits< tflite::SequenceRNNOptions > Struct Reference 152
8.108.1 Detailed Description 152
8.109 tflite::BuiltinOptionsTraits< tflite::ShapeOptions > Struct Reference 153
8.109.1 Detailed Description 153
8.110 tflite::BuiltinOptionsTraits< tflite::SkipGramOptions > Struct Reference 153
8.110.1 Detailed Description 153
8.111 tflite::BuiltinOptionsTraits< tflite::SliceOptions > Struct Reference 153
8.111.1 Detailed Description 153
8.112 tflite::BuiltinOptionsTraits< tflite::SoftmaxOptions > Struct Reference 154
8.112.1 Detailed Description 154
8.113 tflite::BuiltinOptionsTraits< tflite::SpaceToBatchNDOptions > Struct Reference 154
8.113.1 Detailed Description 154
8.114 tflite::BuiltinOptionsTraits< tflite::SpaceToDepthOptions > Struct Reference 154
8.114.1 Detailed Description 154
8.115 tflite::BuiltinOptionsTraits< tflite::SparseToDenseOptions > Struct Reference 155
8.115.1 Detailed Description 155
8.116 tflite::BuiltinOptionsTraits< tflite::SplitOptions > Struct Reference 155
8.116.1 Detailed Description 155
8.117 tflite::BuiltinOptionsTraits< tflite::SplitVOptions > Struct Reference 155
8.117.1 Detailed Description 155
8.118 tflite::BuiltinOptionsTraits< tflite::SquaredDifferenceOptions > Struct Reference 156
8.118.1 Detailed Description 156
8.119 tflite::BuiltinOptionsTraits< tflite::SquareOptions > Struct Reference 156
8.119.1 Detailed Description 156
8.120 tflite::BuiltinOptionsTraits< tflite::SqueezeOptions > Struct Reference 156
8.120.1 Detailed Description 156

8.121 tflite::BuiltinOptionsTraits< tflite::StridedSliceOptions > Struct Reference . . . 157
8.121.1 Detailed Description . . . 157
8.122 tflite::BuiltinOptionsTraits< tflite::SubOptions > Struct Reference . . . 157
8.122.1 Detailed Description . . . 157
8.123 tflite::BuiltinOptionsTraits< tflite::SVDFOptions > Struct Reference . . . 157
8.123.1 Detailed Description . . . 157
8.124 tflite::BuiltinOptionsTraits< tflite::TileOptions > Struct Reference . . . 158
8.124.1 Detailed Description . . . 158
8.125 tflite::BuiltinOptionsTraits< tflite::TopKV2Options > Struct Reference . . . 158
8.125.1 Detailed Description . . . 158
8.126 tflite::BuiltinOptionsTraits< tflite::TransposeConvOptions > Struct Reference . . . 158
8.126.1 Detailed Description . . . 158
8.127 tflite::BuiltinOptionsTraits< tflite::TransposeOptions > Struct Reference . . . 159
8.127.1 Detailed Description . . . 159
8.128 tflite::BuiltinOptionsTraits< tflite::UnidirectionalSequenceLSTMOptions > Struct Reference . . . 159
8.128.1 Detailed Description . . . 159
8.129 tflite::BuiltinOptionsTraits< tflite::UniqueOptions > Struct Reference . . . 159
8.129.1 Detailed Description . . . 159
8.130 tflite::BuiltinOptionsTraits< tflite::UnpackOptions > Struct Reference . . . 160
8.130.1 Detailed Description . . . 160
8.131 tflite::BuiltinOptionsTraits< tflite::WhereOptions > Struct Reference . . . 160
8.131.1 Detailed Description . . . 160
8.132 tflite::BuiltinOptionsTraits< tflite::WhileOptions > Struct Reference . . . 160
8.132.1 Detailed Description . . . 160
8.133 tflite::BuiltinOptionsTraits< tflite::ZerosLikeOptions > Struct Reference . . . 161
8.133.1 Detailed Description . . . 161
8.134 nntrainer::CacheElem Class Reference . . . 161
8.134.1 Detailed Description . . . 162
8.134.2 Member Enumeration Documentation . . . 162
8.134.2.1 Options . . . 162
8.134.3 Constructor & Destructor Documentation . . . 162
8.134.3.1 CacheElem() . . . 162
8.134.3.2 ~CacheElem() . . . 162
8.134.4 Member Function Documentation . . . 163
8.134.4.1 getId() . . . 163
8.134.4.2 getLength() . . . 163
8.134.4.3 isActive() . . . 163
8.134.4.4 reset() . . . 163
8.134.4.5 swapIn() . . . 163
8.134.4.6 swapOut() . . . 164
8.135 nntrainer::CacheLoader Class Reference . . . 164
8.135.1 Detailed Description . . . 165

8.135.2 Constructor & Destructor Documentation . . . 165
8.135.2.1 CacheLoader() . . . 165
8.135.2.2 ~CacheLoader() . . . 165
8.135.3 Member Function Documentation . . . 165
8.135.3.1 cancelAsync() . . . 165
8.135.3.2 finish() . . . 166
8.135.3.3 load() . . . 166
8.135.3.4 loadAsync() [1/2] . . . 166
8.135.3.5 loadAsync() [2/2] . . . 167
8.136 nntrainer::CachePool Class Reference . . . 168
8.136.1 Detailed Description . . . 170
8.136.2 Member Typedef Documentation . . . 170
8.136.2.1 CacheElems . . . 170
8.136.3 Constructor & Destructor Documentation . . . 170
8.136.3.1 CachePool() [1/2] . . . 170
8.136.3.2 CachePool() [2/2] . . . 170
8.136.3.3 ~CachePool() . . . 171
8.136.4 Member Function Documentation . . . 171
8.136.4.1 allocate() . . . 171
8.136.4.2 clear() . . . 171
8.136.4.3 deallocate() . . . 172
8.136.4.4 flush() . . . 172
8.136.4.5 flushExcept() [1/2] . . . 173
8.136.4.6 flushExcept() [2/2] . . . 174
8.136.4.7 getCachePolicy() . . . 174
8.136.4.8 getMemory() . . . 174
8.136.4.9 getName() . . . 175
8.136.4.10 initCacheElemIter() . . . 175
8.136.4.11 initExecIdsIter() . . . 175
8.136.4.12 invalidate() . . . 176
8.136.4.13 isAllocated() . . . 176
8.136.4.14 isLastCacheElemIter() . . . 177
8.136.4.15 isLastExecIdsIter() . . . 177
8.136.4.16 loadExec() . . . 177
8.136.4.17 loadExecOnce() . . . 178
8.136.4.18 requestMemory() . . . 178
8.136.4.19 unloadExec() . . . 179
8.136.4.20 validate() . . . 180
8.137 tflite::CallOptionsBuilder Struct Reference . . . 180
8.137.1 Detailed Description . . . 181
8.138 tflite::CastOptionsBuilder Struct Reference . . . 181
8.138.1 Detailed Description . . . 181

8.139 nntrainer::CentroidKNN Class Reference . . . 182
8.139.1 Detailed Description . . . 183
8.139.2 Constructor & Destructor Documentation . . . 183
8.139.2.1 CentroidKNN() . . . 183
8.139.2.2 ~CentroidKNN() . . . 183
8.139.3 Member Function Documentation . . . 183
8.139.3.1 calcDerivative() . . . 184
8.139.3.2 exportTo() . . . 184
8.139.3.3 finalize() . . . 184
8.139.3.4 forwarding() . . . 185
8.139.3.5 getType() . . . 185
8.139.3.6 requireLabel() . . . 185
8.139.3.7 setProperty() . . . 186
8.139.3.8 supportBackwarding() . . . 186
8.140 nntrainer::props::ClipGradByGlobalNorm Class Reference . . . 186
8.140.1 Detailed Description . . . 187
8.140.2 Member Typedef Documentation . . . 188
8.140.2.1 prop_tag . . . 188
8.140.3 Member Data Documentation . . . 188
8.140.3.1 key . . . 188
8.141 nntrainer::props::ConcatDimension Class Reference . . . 188
8.141.1 Detailed Description . . . 190
8.142 tflite::ConcatEmbeddingsOptionsBuilder Struct Reference . . . 190
8.142.1 Detailed Description . . . 190
8.143 tflite::ConcatenationOptionsBuilder Struct Reference . . . 190
8.143.1 Detailed Description . . . 191
8.144 nntrainer::Connection Class Reference . . . 191
8.144.1 Detailed Description . . . 192
8.144.2 Constructor & Destructor Documentation . . . 192
8.144.2.1 Connection() [1/4] . . . 192
8.144.2.2 Connection() [2/4] . . . 192
8.144.2.3 Connection() [3/4] . . . 193
8.144.2.4 Connection() [4/4] . . . 193
8.144.3 Member Function Documentation . . . 193
8.144.3.1 getIndex() [1/2] . . . 193
8.144.3.2 getIndex() [2/2] . . . 194
8.144.3.3 getName() [1/2] . . . 194
8.144.3.4 getName() [2/2] . . . 194
8.144.3.5 operator=() [1/2] . . . 195
8.144.3.6 operator=() [2/2] . . . 195
8.144.3.7 operator==() . . . 195
8.144.3.8 toString() . . . 196

8.145 nntrainer::props::connection_prop_tag Struct Reference . . . 197
8.145.1 Detailed Description . . . 197
8.146 tflite::Conv2DOptionsBuilder Struct Reference . . . 197
8.146.1 Detailed Description . . . 197
8.147 tflite::CosOptionsBuilder Struct Reference . . . 198
8.147.1 Detailed Description . . . 198
8.148 tflite::CumsumOptionsBuilder Struct Reference . . . 198
8.148.1 Detailed Description . . . 199
8.149 tflite::CustomQuantizationBuilder Struct Reference . . . 199
8.149.1 Detailed Description . . . 199
8.150 nntrainer::DataProducer Class Reference . . . 199
8.150.1 Detailed Description . . . 200
8.150.2 Member Typedef Documentation . . . 200
8.150.2.1 Generator . . . 200
8.150.3 Constructor & Destructor Documentation . . . 201
8.150.3.1 ∼DataProducer() . . . 201
8.150.4 Member Function Documentation . . . 201
8.150.4.1 exportTo() . . . 201
8.150.4.2 finalize() . . . 202
8.150.4.3 getType() . . . 202
8.150.4.4 isMultiThreadSafe() . . . 203
8.150.4.5 setProperty() . . . 203
8.150.4.6 size() . . . 203
8.150.5 Member Data Documentation . . . 204
8.150.5.1 SIZE_UNDEFINED . . . 204
8.151 nntrainer::props::DecayRate Class Reference . . . 204
8.151.1 Detailed Description . . . 205
8.151.2 Member Typedef Documentation . . . 205
8.151.2.1 prop_tag . . . 205
8.151.3 Member Data Documentation . . . 205
8.151.3.1 key . . . 206
8.152 nntrainer::props::DecaySteps Class Reference . . . 206
8.152.1 Detailed Description . . . 207
8.152.2 Member Typedef Documentation . . . 207
8.152.2.1 prop_tag . . . 207
8.152.3 Member Data Documentation . . . 207
8.152.3.1 key . . . 207
8.153 nntrainer::DelegateConfig Class Reference . . . 208
8.153.1 Detailed Description . . . 208
8.153.2 Constructor & Destructor Documentation . . . 208
8.153.2.1 DelegateConfig() . . . 208
8.154 tflite::DensifyOptionsBuilder Struct Reference . . . 208

8.154.1 Detailed Description . . . 209
8.155 tflite::DepthToSpaceOptionsBuilder Struct Reference . . . 209
8.155.1 Detailed Description . . . 209
8.156 tflite::DepthwiseConv2DOptionsBuilder Struct Reference . . . 210
8.156.1 Detailed Description . . . 210
8.157 tflite::DequantizeOptionsBuilder Struct Reference . . . 210
8.157.1 Detailed Description . . . 211
8.158 nntrainer::Device Class Reference . . . 211
8.158.1 Detailed Description . . . 211
8.158.2 Member Enumeration Documentation . . . 212
8.158.2.1 DeviceType . . . 212
8.158.3 Constructor & Destructor Documentation . . . 212
8.158.3.1 Device() . . . 212
8.158.4 Member Function Documentation . . . 212
8.158.4.1 allowPrecisionLoss() . . . 213
8.158.4.2 allowSoftPlacement() . . . 213
8.158.4.3 getDevice() . . . 213
8.158.4.4 setDevice() . . . 214
8.158.4.5 setMemoryFraction() . . . 214
8.159 nntrainer::props::Dilation Class Reference . . . 214
8.159.1 Detailed Description . . . 215
8.159.2 Member Typedef Documentation . . . 215
8.159.2.1 prop_tag . . . 215
8.159.3 Constructor & Destructor Documentation . . . 216
8.159.3.1 Dilation() . . . 216
8.159.4 Member Data Documentation . . . 216
8.159.4.1 key . . . 216
8.160 tflite::DimensionMetadataBuilder Struct Reference . . . 216
8.160.1 Detailed Description . . . 217
8.161 nntrainer::DirDataProducer Class Reference . . . 217
8.161.1 Detailed Description . . . 218
8.161.2 Constructor & Destructor Documentation . . . 218
8.161.2.1 DirDataProducer() [1/2] . . . 218
8.161.2.2 DirDataProducer() [2/2] . . . 219
8.161.2.3 ~DirDataProducer() . . . 219
8.161.3 Member Function Documentation . . . 219
8.161.3.1 finalize() . . . 219
8.161.3.2 getType() . . . 219
8.161.3.3 isMultiThreadSafe() . . . 220
8.161.3.4 setProperty() . . . 220
8.161.3.5 size() . . . 221
8.162 nntrainer::props::DirPath Class Reference . . . 221

8.162.1 Detailed Description . . . 222
8.162.2 Member Typedef Documentation . . . 222
8.162.2.1 prop_tag . . . 222
8.162.3 Constructor & Destructor Documentation . . . 222
8.162.3.1 DirPath() . . . 222
8.162.4 Member Function Documentation . . . 223
8.162.4.1 isValid() . . . 223
8.162.4.2 set() . . . 223
8.162.5 Member Data Documentation . . . 224
8.162.5.1 key . . . 224
8.163 nntrainer::props::DisableBias Class Reference . . . 224
8.163.1 Detailed Description . . . 225
8.163.2 Constructor & Destructor Documentation . . . 226
8.163.2.1 DisableBias() . . . 226
8.164 nntrainer::props::Distribute Class Reference . . . 226
8.164.1 Detailed Description . . . 227
8.165 tflite::DivOptionsBuilder Struct Reference . . . 227
8.165.1 Detailed Description . . . 227
8.166 nntrainer::props::DropOutRate Class Reference . . . 228
8.166.1 Detailed Description . . . 229
8.166.2 Member Typedef Documentation . . . 229
8.166.2.1 prop_tag . . . 229
8.166.3 Constructor & Destructor Documentation . . . 229
8.166.3.1 DropOutRate() . . . 229
8.166.4 Member Function Documentation . . . 229
8.166.4.1 isValid() . . . 229
8.166.5 Member Data Documentation . . . 230
8.166.5.1 key . . . 230
8.167 nntrainer::props::DynamicTimeSequence Class Reference . . . 230
8.167.1 Detailed Description . . . 231
8.167.2 Constructor & Destructor Documentation . . . 231
8.167.2.1 DynamicTimeSequence() . . . 231
8.168 tflite::EmbeddingLookupSparseOptionsBuilder Struct Reference . . . 232
8.168.1 Detailed Description . . . 232
8.169 nntrainer::props::Epsilon Class Reference . . . 232
8.169.1 Detailed Description . . . 233
8.169.2 Member Typedef Documentation . . . 233
8.169.2.1 prop_tag . . . 233
8.169.3 Constructor & Destructor Documentation . . . 234
8.169.3.1 Epsilon() . . . 234
8.169.4 Member Function Documentation . . . 234
8.169.4.1 isValid() . . . 234

8.169.5 Member Data Documentation . . . 234
8.169.5.1 key . . . 234
8.170 tflite::EqualOptionsBuilder Struct Reference . . . 235
8.170.1 Detailed Description . . . 235
8.171 nntrainer::exception::ErrorNotification< Err, > Class Template Reference . . . 235
8.171.1 Detailed Description . . . 235
8.171.2 Constructor & Destructor Documentation . . . 236
8.171.2.1 ErrorNotification() [1/2] . . . 236
8.171.2.2 ErrorNotification() [2/2] . . . 236
8.171.2.3 ~ErrorNotification() . . . 236
8.171.3 Friends And Related Function Documentation . . . 237
8.171.3.1 operator<< . . . 237
8.172 tflite::ExpandDimsOptionsBuilder Struct Reference . . . 237
8.172.1 Detailed Description . . . 238
8.173 tflite::ExpOptionsBuilder Struct Reference . . . 238
8.173.1 Detailed Description . . . 238
8.174 nntrainer::Exporter Class Reference . . . 238
8.174.1 Detailed Description . . . 239
8.174.2 Constructor & Destructor Documentation . . . 239
8.174.2.1 Exporter() . . . 239
8.174.2.2 ~Exporter() . . . 239
8.174.3 Member Function Documentation . . . 239
8.174.3.1 getResult() . . . 239
8.174.3.2 saveResult() . . . 240
8.175 External Struct Reference . . . 241
8.175.1 Detailed Description . . . 242
8.176 tflite::FakeQuantOptionsBuilder Struct Reference . . . 242
8.176.1 Detailed Description . . . 242
8.177 nntrainer::props::FilePath Class Reference . . . 243
8.177.1 Detailed Description . . . 244
8.177.2 Member Typedef Documentation . . . 244
8.177.2.1 prop_tag . . . 244
8.177.3 Constructor & Destructor Documentation . . . 244
8.177.3.1 FilePath() . . . 244
8.177.4 Member Function Documentation . . . 245
8.177.4.1 file_size() . . . 245
8.177.4.2 isValid() . . . 245
8.177.4.3 set() . . . 246
8.177.5 Member Data Documentation . . . 246
8.177.5.1 key . . . 246
8.178 tflite::FillOptionsBuilder Struct Reference . . . 246
8.178.1 Detailed Description . . . 247

8.179 nntrainer::props::FilterSize Class Reference . . . 247
8.179.1 Detailed Description . . . 248
8.179.2 Member Typedef Documentation . . . 248
8.179.2.1 prop_tag . . . 248
8.179.3 Member Data Documentation . . . 248
8.179.3.1 key . . . 248
8.180 nntrainer::FlatBufferInterpreter Class Reference . . . 249
8.180.1 Detailed Description . . . 249
8.180.2 Constructor & Destructor Documentation . . . 250
8.180.2.1 FlatBufferInterpreter() . . . 250
8.180.2.2 ∼FlatBufferInterpreter() . . . 250
8.180.3 Member Function Documentation . . . 250
8.180.3.1 deserialize() . . . 250
8.180.3.2 serialize() . . . 251
8.181 nntrainer::FlatBufferOpNode Class Reference . . . 251
8.181.1 Detailed Description . . . 252
8.181.2 Constructor & Destructor Documentation . . . 252
8.181.2.1 FlatBufferOpNode() . . . 252
8.181.3 Member Function Documentation . . . 252
8.181.3.1 arg() . . . 252
8.181.3.2 arity() [1/2] . . . 253
8.181.3.3 arity() [2/2] . . . 253
8.181.3.4 getBuiltinOps() . . . 253
8.181.3.5 getInputNodes() . . . 254
8.181.3.6 getInputs() [1/2] . . . 254
8.181.3.7 getInputs() [2/2] . . . 254
8.181.3.8 getOptionType() . . . 255
8.181.3.9 getOpType() . . . 255
8.181.3.10 getOutputs() [1/2] . . . 255
8.181.3.11 getOutputs() [2/2] . . . 255
8.181.3.12 getWeights() [1/2] . . . 256
8.181.3.13 getWeights() [2/2] . . . 256
8.181.3.14 isInputNode() . . . 256
8.181.3.15 isOutputNode() . . . 256
8.181.3.16 isVirtualNode() . . . 257
8.181.3.17 setArg() . . . 257
8.181.3.18 setBuiltinOptions() . . . 257
8.181.3.19 setLayerNode() . . . 258
8.181.3.20 setOpType() . . . 258
8.182 tflite::FLATBUFFERS_FINAL_CLASS Struct Reference . . . 259
8.182.1 Detailed Description . . . 270
8.183 nntrainer::props::Flatten Class Reference . . . 270

8.183.1 Detailed Description . . . 271
8.183.2 Member Typedef Documentation . . . 271
8.183.2.1 prop_tag . . . 271
8.183.3 Member Data Documentation . . . 271
8.183.3.1 key . . . 271
8.184 nntrainer::FlattenRealizer Class Reference . . . 272
8.184.1 Detailed Description . . . 272
8.184.2 Constructor & Destructor Documentation . . . 273
8.184.2.1 ~FlattenRealizer() . . . 273
8.184.3 Member Function Documentation . . . 273
8.184.3.1 realize() . . . 273
8.185 nntrainer::props::FlipDirection Class Reference . . . 274
8.185.1 Detailed Description . . . 275
8.186 nntrainer::props::FlipDirectionInfo Struct Reference . . . 275
8.186.1 Detailed Description . . . 275
8.186.2 Member Data Documentation . . . 275
8.186.2.1 EnumList . . . 275
8.186.2.2 EnumStr . . . 276
8.187 tflite::FloorDivOptionsBuilder Struct Reference . . . 276
8.187.1 Detailed Description . . . 276
8.188 tflite::FloorModOptionsBuilder Struct Reference . . . 276
8.188.1 Detailed Description . . . 277
8.189 tflite::FullyConnectedOptionsBuilder Struct Reference . . . 277
8.189.1 Detailed Description . . . 277
8.190 nntrainer::FuncDataProducer Class Reference . . . 278
8.190.1 Detailed Description . . . 279
8.190.2 Constructor & Destructor Documentation . . . 279
8.190.2.1 FuncDataProducer() . . . 279
8.190.2.2 ~FuncDataProducer() . . . 279
8.190.3 Member Function Documentation . . . 279
8.190.3.1 exportTo() . . . 280
8.190.3.2 finalize() . . . 280
8.190.3.3 getType() . . . 280
8.190.3.4 setProperty() . . . 281
8.191 tflite::GatherNdOptionsBuilder Struct Reference . . . 281
8.191.1 Detailed Description . . . 281
8.192 tflite::GatherOptionsBuilder Struct Reference . . . 282
8.192.1 Detailed Description . . . 282
8.193 nntrainer::profile::GenericProfileListener Class Reference . . . 282
8.193.1 Detailed Description . . . 283
8.193.2 Constructor & Destructor Documentation . . . 283
8.193.2.1 GenericProfileListener() . . . 283

8.193.2.2 ~GenericProfileListener() . . . 284
8.193.3 Member Function Documentation . . . 284
8.193.3.1 notify() . . . 284
8.193.3.2 report() . . . 284
8.193.3.3 reset() . . . 285
8.193.3.4 result() . . . 285
8.194 nntrainer::props::GenericShape Class Reference . . . 285
8.194.1 Detailed Description . . . 287
8.194.2 Member Typedef Documentation . . . 287
8.194.2.1 prop_tag . . . 287
8.194.3 Member Function Documentation . . . 287
8.194.3.1 set() . . . 287
8.194.4 Member Data Documentation . . . 287
8.194.4.1 key . . . 288
8.195 nntrainer::GraphCompiler Class Reference . . . 288
8.195.1 Detailed Description . . . 288
8.195.2 Member Function Documentation . . . 288
8.195.2.1 compile() . . . 288
8.195.2.2 decompile() . . . 289
8.196 nntrainer::GraphInterpreter Class Reference . . . 289
8.196.1 Detailed Description . . . 290
8.196.2 Member Function Documentation . . . 290
8.196.2.1 deserialize() . . . 290
8.196.2.2 deserialize_v1() . . . 290
8.196.2.3 serialize() . . . 291
8.196.2.4 serialize_v1() . . . 291
8.197 nntrainer::GraphNode Class Reference . . . 292
8.197.1 Detailed Description . . . 292
8.197.2 Member Typedef Documentation . . . 293
8.197.2.1 ExecutionOrder . . . 293
8.197.3 Member Function Documentation . . . 293
8.197.3.1 getExecutionOrder() . . . 293
8.197.3.2 getInputConnections() . . . 293
8.197.3.3 getName() . . . 294
8.197.3.4 getOutputConnections() . . . 294
8.197.3.5 getTrainable() . . . 294
8.197.3.6 getType() . . . 295
8.197.3.7 setExecutionOrder() . . . 295
8.197.3.8 setName() . . . 295
8.198 nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType > Class Template Reference . . . 296
8.198.1 Detailed Description . . . 297
8.198.2 Constructor & Destructor Documentation . . . 298

8.198.2.1 GraphNodeIterator() . . . 298
8.198.3 Member Function Documentation . . . 298
8.198.3.1 operator∗() . . . 298
8.198.3.2 operator+() . . . 298
8.198.3.3 operator++() [1/2] . . . 299
8.198.3.4 operator++() [2/2] . . . 299
8.198.3.5 operator+=() . . . 299
8.198.3.6 operator-() [1/2] . . . 300
8.198.3.7 operator-() [2/2] . . . 300
8.198.3.8 operator--() [1/2] . . . 301
8.198.3.9 operator--() [2/2] . . . 301
8.198.3.10 operator-=() . . . 301
8.198.3.11 operator->() . . . 302
8.198.4 Friends And Related Function Documentation . . . 302
8.198.4.1 operator"!= . . . 302
8.198.4.2 operator== . . . 303
8.198.5 Member Data Documentation . . . 303
8.198.5.1 value_type . . . 303
8.199 nntrainer::GraphNodeReverseIterator< T_iterator > Class Template Reference . . . 304
8.199.1 Detailed Description . . . 305
8.199.2 Constructor & Destructor Documentation . . . 305
8.199.2.1 GraphNodeReverseIterator() . . . 305
8.199.3 Member Function Documentation . . . 305
8.199.3.1 operator∗() . . . 305
8.199.3.2 operator->() . . . 306
8.200 nntrainer::GraphRealizer Class Reference . . . 306
8.200.1 Detailed Description . . . 307
8.200.2 Constructor & Destructor Documentation . . . 307
8.200.2.1 ∼GraphRealizer() . . . 308
8.200.3 Member Function Documentation . . . 308
8.200.3.1 realize() . . . 308
8.201 tflite::GreaterEqualOptionsBuilder Struct Reference . . . 308
8.201.1 Detailed Description . . . 309
8.202 tflite::GreaterOptionsBuilder Struct Reference . . . 309
8.202.1 Detailed Description . . . 309
8.203 tflite::HardSwishOptionsBuilder Struct Reference . . . 309
8.203.1 Detailed Description . . . 310
8.204 std::hash< nntrainer::Connection > Struct Reference . . . 310
8.204.1 Detailed Description . . . 310
8.204.2 Member Function Documentation . . . 310
8.204.2.1 operator()() . . . 310
8.205 nntrainer::props::HiddenStateActivation Class Reference . . . 311

8.205.1 Detailed Description . . . 312
8.205.2 Constructor & Destructor Documentation . . . 312
8.205.2.1 HiddenStateActivation() . . . 312
8.206 tflite::IfOptionsBuilder Struct Reference . . . 313
8.206.1 Detailed Description . . . 313
8.207 nntrainer::props::InDim Class Reference . . . 313
8.207.1 Detailed Description . . . 314
8.207.2 Member Typedef Documentation . . . 314
8.207.2.1 prop_tag . . . 314
8.207.3 Member Data Documentation . . . 314
8.207.3.1 key . . . 315
8.208 nntrainer::IniGraphInterpreter Class Reference . . . 315
8.208.1 Detailed Description . . . 316
8.208.2 Constructor & Destructor Documentation . . . 316
8.208.2.1 IniGraphInterpreter() . . . 316
8.208.2.2 ∼IniGraphInterpreter() . . . 316
8.208.3 Member Function Documentation . . . 317
8.208.3.1 deserialize() . . . 317
8.208.3.2 serialize() . . . 318
8.209 nntrainer::IniSection Class Reference . . . 318
8.209.1 Detailed Description . . . 319
8.209.2 Constructor & Destructor Documentation . . . 319
8.209.2.1 IniSection() [1/5] . . . 319
8.209.2.2 IniSection() [2/5] . . . 320
8.209.2.3 IniSection() [3/5] . . . 320
8.209.2.4 IniSection() [4/5] . . . 321
8.209.2.5 IniSection() [5/5] . . . 321
8.209.2.6 ∼IniSection() . . . 322
8.209.3 Member Function Documentation . . . 322
8.209.3.1 FromExportable() . . . 322
8.209.3.2 getName() . . . 323
8.209.3.3 operator"!=() . . . 323
8.209.3.4 operator+() [1/2] . . . 324
8.209.3.5 operator+() [2/2] . . . 325
8.209.3.6 operator+=() [1/2] . . . 325
8.209.3.7 operator+=() [2/2] . . . 326
8.209.3.8 operator==() . . . 327
8.209.3.9 print() . . . 328
8.209.3.10 setEntry() . . . 328
8.209.4 Friends And Related Function Documentation . . . 328
8.209.4.1 operator<< . . . 329
8.210 nntrainer::props::InitializerInfo Struct Reference . . . 329

8.210.1 Detailed Description . . . 329
8.210.2 Member Data Documentation . . . 330
8.210.2.1 EnumList . . . 330
8.210.2.2 EnumStr . . . 330
8.211 nntrainer::InitLayerContext Class Reference . . . 330
8.211.1 Detailed Description . . . 332
8.211.2 Member Typedef Documentation . . . 332
8.211.2.1 TensorSpec . . . 332
8.211.3 Constructor & Destructor Documentation . . . 332
8.211.3.1 InitLayerContext() . . . 332
8.211.4 Member Function Documentation . . . 332
8.211.4.1 executeInPlace() . . . 333
8.211.4.2 getInputDimensions() . . . 333
8.211.4.3 getName() . . . 333
8.211.4.4 getNumInputs() . . . 334
8.211.4.5 getNumRequestedOutputs() . . . 334
8.211.4.6 getNumTensors() . . . 334
8.211.4.7 getNumWeights() . . . 334
8.211.4.8 getOutSpecs() . . . 335
8.211.4.9 getTensorsSpec() . . . 335
8.211.4.10 getWeightsSpec() . . . 335
8.211.4.11 outSpec() . . . 335
8.211.4.12 requestOutputs() . . . 336
8.211.4.13 requestTensor() [1/2] . . . 337
8.211.4.14 requestTensor() [2/2] . . . 337
8.211.4.15 requestWeight() [1/2] . . . 338
8.211.4.16 requestWeight() [2/2] . . . 338
8.211.4.17 setDynDimFlagInputDimension() . . . 339
8.211.4.18 setEffDimFlagInputDimension() . . . 339
8.211.4.19 setOutputDimensions() . . . 339
8.211.4.20 validate() . . . 340
8.212 nntrainer::IniWrapper Class Reference . . . 341
8.212.1 Detailed Description . . . 342
8.212.2 Constructor & Destructor Documentation . . . 342
8.212.2.1 IniWrapper() [1/2] . . . 342
8.212.2.2 IniWrapper() [2/2] . . . 342
8.212.3 Member Function Documentation . . . 343
8.212.3.1 erase_ini() . . . 343
8.212.3.2 getIniName() . . . 343
8.212.3.3 getName() . . . 344
8.212.3.4 operator"!=() . . . 344
8.212.3.5 operator+() [1/3] . . . 344

8.212.3.6 operator+() [2/3] . . . 346
8.212.3.7 operator+() [3/3] . . . 347
8.212.3.8 operator+=() [1/3] . . . 348
8.212.3.9 operator+=() [2/3] . . . 348
8.212.3.10 operator+=() [3/3] . . . 348
8.212.3.11 operator==() . . . 349
8.212.3.12 save_ini() . . . 349
8.212.4 Friends And Related Function Documentation . . . 350
8.212.4.1 operator<< . . . 350
8.213 nntrainer::props::InputConnection Class Reference . . . 350
8.213.1 Detailed Description . . . 351
8.213.2 Member Typedef Documentation . . . 351
8.213.2.1 prop_tag . . . 351
8.213.3 Constructor & Destructor Documentation . . . 352
8.213.3.1 InputConnection() [1/2] . . . 352
8.213.3.2 InputConnection() [2/2] . . . 352
8.213.4 Member Data Documentation . . . 352
8.213.4.1 key . . . 352
8.214 nntrainer::props::InputIsSequence Class Reference . . . 353
8.214.1 Detailed Description . . . 354
8.215 nntrainer::InputRealizer Class Reference . . . 354
8.215.1 Detailed Description . . . 355
8.215.2 Constructor & Destructor Documentation . . . 355
8.215.2.1 InputRealizer() . . . 355
8.215.2.2 ∼InputRealizer() . . . 355
8.215.3 Member Function Documentation . . . 356
8.215.3.1 realize() . . . 356
8.216 nntrainer::props::InputShape Class Reference . . . 356
8.216.1 Detailed Description . . . 358
8.216.2 Member Typedef Documentation . . . 358
8.216.2.1 prop_tag . . . 358
8.216.3 Member Data Documentation . . . 358
8.216.3.1 key . . . 358
8.217 tflite::Int32VectorBuilder Struct Reference . . . 358
8.217.1 Detailed Description . . . 359
8.218 nntrainer::props::IntegrateBias Class Reference . . . 359
8.218.1 Detailed Description . . . 360
8.218.2 Constructor & Destructor Documentation . . . 360
8.218.2.1 IntegrateBias() . . . 360
8.219 nntrainer::Iteration Class Reference . . . 361
8.219.1 Detailed Description . . . 361
8.219.2 Constructor & Destructor Documentation . . . 361

8.219.2.1 Iteration() . . . 362
8.219.3 Member Function Documentation . . . 362
8.219.3.1 batch() . . . 362
8.219.3.2 begin() [1/2] . . . 363
8.219.3.3 begin() [2/2] . . . 363
8.219.3.4 end() [1/2] . . . 364
8.219.3.5 end() [2/2] . . . 364
8.219.3.6 getInputsRef() [1/2] . . . 364
8.219.3.7 getInputsRef() [2/2] . . . 365
8.219.3.8 getLabelsRef() [1/2] . . . 365
8.219.3.9 getLabelsRef() [2/2] . . . 365
8.219.3.10 setEndSample() [1/2] . . . 365
8.219.3.11 setEndSample() [2/2] . . . 366
8.220 nntrainer::props::Iteration Class Reference . . . 366
8.220.1 Detailed Description . . . 367
8.220.2 Member Typedef Documentation . . . 367
8.220.2.1 prop_tag . . . 367
8.220.3 Member Data Documentation . . . 367
8.220.3.1 key . . . 367
8.221 nntrainer::IterationQueue Class Reference . . . 368
8.221.1 Detailed Description . . . 368
8.221.2 Constructor & Destructor Documentation . . . 369
8.221.2.1 IterationQueue() . . . 369
8.221.2.2 ∼IterationQueue() . . . 369
8.221.3 Member Function Documentation . . . 370
8.221.3.1 batch() . . . 370
8.221.3.2 notifyEndOfRequestEmpty() . . . 370
8.221.3.3 requestEmptySlot() . . . 370
8.221.3.4 requestFilledSlot() . . . 371
8.221.3.5 slots() . . . 371
8.222 nntrainer::props::KernelSize Class Reference . . . 372
8.222.1 Detailed Description . . . 372
8.222.2 Member Typedef Documentation . . . 373
8.222.2.1 prop_tag . . . 373
8.222.3 Member Data Documentation . . . 373
8.222.3.1 key . . . 373
8.223 tflite::L2NormOptionsBuilder Struct Reference . . . 373
8.223.1 Detailed Description . . . 374
8.224 nntrainer::props::LabelLayer Class Reference . . . 374
8.224.1 Detailed Description . . . 375
8.224.2 Constructor & Destructor Documentation . . . 375
8.224.2.1 LabelLayer() [1/2] . . . 375

8.224.2.2 LabelLayer() [2/2] . . . 375
8.225 Layer Class Reference . . . 376
8.225.1 Detailed Description . . . 376
8.226 Layer Class Reference . . . 376
8.226.1 Detailed Description . . . 376
8.227 Layer Class Reference . . . 377
8.227.1 Detailed Description . . . 377
8.228 nntrainer::LayerNode Class Reference . . . 377
8.228.1 Detailed Description . . . 381
8.228.2 Member Enumeration Documentation . . . 381
8.228.2.1 PrintPreset . . . 381
8.228.3 Constructor & Destructor Documentation . . . 381
8.228.3.1 LayerNode() . . . 381
8.228.3.2 ∼LayerNode() . . . 382
8.228.4 Member Function Documentation . . . 382
8.228.4.1 calcDerivative() . . . 382
8.228.4.2 calcGradient() . . . 382
8.228.4.3 clearOptVar() . . . 382
8.228.4.4 cloneConfiguration() . . . 383
8.228.4.5 configureRunContext() . . . 383
8.228.4.6 executeInPlace() [1/2] . . . 383
8.228.4.7 executeInPlace() [2/2] . . . 384
8.228.4.8 exportTo() . . . 384
8.228.4.9 finalize() . . . 385
8.228.4.10 forwarding() . . . 385
8.228.4.11 getActivationToBeRealized() . . . 386
8.228.4.12 getActivationType() . . . 386
8.228.4.13 getDistribute() . . . 387
8.228.4.14 getExecutionOrder() . . . 387
8.228.4.15 getFlatten() . . . 388
8.228.4.16 getInput() . . . 388
8.228.4.17 getInputConnectionIndex() . . . 388
8.228.4.18 getInputConnectionName() . . . 389
8.228.4.19 getInputConnections() . . . 389
8.228.4.20 getInputDimensions() . . . 390
8.228.4.21 getInputGrad() . . . 390
8.228.4.22 getLoss() . . . 390
8.228.4.23 getName() . . . 390
8.228.4.24 getNumInputConnections() . . . 391
8.228.4.25 getNumInputs() . . . 392
8.228.4.26 getNumOutputConnections() . . . 392
8.228.4.27 getNumOutputs() . . . 393

8.228.4.28 getNumWeights() . . . 393
8.228.4.29 getOutput() . . . 393
8.228.4.30 getOutputConnection() . . . 394
8.228.4.31 getOutputConnections() . . . 394
8.228.4.32 getOutputDimensions() . . . 395
8.228.4.33 getOutputGrad() . . . 395
8.228.4.34 getOutputGradUnsafe() . . . 396
8.228.4.35 getRunContext() [1/2] . . . 396
8.228.4.36 getRunContext() [2/2] . . . 397
8.228.4.37 getSharedFrom() . . . 397
8.228.4.38 getTrainable() . . . 398
8.228.4.39 getType() . . . 398
8.228.4.40 getWeight() . . . 398
8.228.4.41 getWeightGrad() . . . 399
8.228.4.42 getWeightName() . . . 399
8.228.4.43 getWeightObject() . . . 400
8.228.4.44 getWeights() [1/2] . . . 400
8.228.4.45 getWeights() [2/2] . . . 400
8.228.4.46 getWeightWrapper() . . . 401
8.228.4.47 hasInputShapeProperty() . . . 401
8.228.4.48 isFinalized() . . . 402
8.228.4.49 needsCalcDerivative() [1/2] . . . 402
8.228.4.50 needsCalcDerivative() [2/2] . . . 402
8.228.4.51 needsCalcGradient() [1/2] . . . 402
8.228.4.52 needsCalcGradient() [2/2] . . . 403
8.228.4.53 printPreset() . . . 403
8.228.4.54 read() . . . 403
8.228.4.55 remapConnections() . . . 404
8.228.4.56 remapIdentifiers() . . . 404
8.228.4.57 requireLabel() . . . 405
8.228.4.58 save() . . . 405
8.228.4.59 setBatch() . . . 405
8.228.4.60 setExecutionOrder() . . . 406
8.228.4.61 setInputConnectionIndex() . . . 406
8.228.4.62 setInputConnectionName() . . . 406
8.228.4.63 setName() . . . 407
8.228.4.64 setOutputConnection() . . . 407
8.228.4.65 setOutputLayers() . . . 408
8.228.4.66 setProperty() . . . 408
8.228.4.67 setWeights() . . . 409
8.228.4.68 supportBackwarding() . . . 410
8.228.4.69 supportInPlace() . . . 410

8.229 tflite::LeakyReluOptionsBuilder Struct Reference . . . 411
8.229.1 Detailed Description . . . 411
8.230 nntrainer::props::LearningRate Class Reference . . . 411
8.230.1 Detailed Description . . . 412
8.230.2 Member Typedef Documentation . . . 412
8.230.2.1 prop_tag . . . 412
8.230.3 Member Data Documentation . . . 412
8.230.3.1 key . . . 413
8.231 tflite::LessEqualOptionsBuilder Struct Reference . . . 413
8.231.1 Detailed Description . . . 413
8.232 tflite::LessOptionsBuilder Struct Reference . . . 413
8.232.1 Detailed Description . . . 414
8.233 tflite::LocalResponseNormalizationOptionsBuilder Struct Reference . . . 414
8.233.1 Detailed Description . . . 414
8.234 tflite::LogicalAndOptionsBuilder Struct Reference . . . 415
8.234.1 Detailed Description . . . 415
8.235 tflite::LogicalNotOptionsBuilder Struct Reference . . . 415
8.235.1 Detailed Description . . . 416
8.236 tflite::LogicalOrOptionsBuilder Struct Reference . . . 416
8.236.1 Detailed Description . . . 416
8.237 tflite::LogSoftmaxOptionsBuilder Struct Reference . . . 416
8.237.1 Detailed Description . . . 417
8.238 nntrainer::props::Loss Class Reference . . . 417
8.238.1 Detailed Description . . . 418
8.238.2 Member Typedef Documentation . . . 418
8.238.2.1 prop_tag . . . 418
8.238.3 Constructor & Destructor Documentation . . . 418
8.238.3.1 Loss() . . . 418
8.238.4 Member Function Documentation . . . 419
8.238.4.1 isValid() . . . 419
8.238.5 Member Data Documentation . . . 420
8.238.5.1 key . . . 420
8.239 nntrainer::LossRealizer Class Reference . . . 420
8.239.1 Detailed Description . . . 421
8.239.2 Constructor & Destructor Documentation . . . 421
8.239.2.1 LossRealizer() . . . 421
8.239.2.2 ∼LossRealizer() . . . 422
8.239.3 Member Function Documentation . . . 422
8.239.3.1 realize() . . . 422
8.240 tflite::LSHProjectionOptionsBuilder Struct Reference . . . 423
8.240.1 Detailed Description . . . 423
8.241 tflite::LSTMOptionsBuilder Struct Reference . . . 423

8.241.1 Detailed Description . . . 424
8.242 tflite::MatrixDiagOptionsBuilder Struct Reference . . . 424
8.242.1 Detailed Description . . . 424
8.243 tflite::MatrixSetDiagOptionsBuilder Struct Reference . . . 424
8.243.1 Detailed Description . . . 425
8.244 tflite::MaximumMinimumOptionsBuilder Struct Reference . . . 425
8.244.1 Detailed Description . . . 425
8.245 nntrainer::props::MaxTimestep Class Reference . . . 426
8.245.1 Detailed Description . . . 426
8.245.2 Member Typedef Documentation . . . 427
8.245.2.1 prop_tag . . . 427
8.245.3 Member Data Documentation . . . 427
8.245.3.1 key . . . 427
8.246 nntrainer::MemoryData< T > Class Template Reference . . . 427
8.246.1 Detailed Description . . . 428
8.246.2 Constructor & Destructor Documentation . . . 428
8.246.2.1 MemoryData() [1/2] . . . 428
8.246.2.2 MemoryData() [2/2] . . . 429
8.247 nntrainer::MemoryPlanner Class Reference . . . 429
8.247.1 Detailed Description . . . 430
8.247.2 Constructor & Destructor Documentation . . . 430
8.247.2.1 ~MemoryPlanner() . . . 430
8.247.3 Member Function Documentation . . . 430
8.247.3.1 getType() . . . 430
8.247.3.2 planLayout() . . . 430
8.248 nntrainer::MemoryPool Class Reference . . . 431
8.248.1 Detailed Description . . . 432
8.248.2 Constructor & Destructor Documentation . . . 432
8.248.2.1 MemoryPool() . . . 433
8.248.2.2 ~MemoryPool() . . . 433
8.248.3 Member Function Documentation . . . 433
8.248.3.1 allocate() . . . 433
8.248.3.2 clear() . . . 434
8.248.3.3 deallocate() . . . 434
8.248.3.4 getMemory() . . . 434
8.248.3.5 isAllocated() . . . 435
8.248.3.6 minMemoryRequirement() . . . 435
8.248.3.7 planLayout() . . . 435
8.248.3.8 requestMemory() . . . 436
8.248.3.9 size() . . . 437
8.249 nntrainer::MemoryRequest Struct Reference . . . 438
8.249.1 Detailed Description . . . 438

8.249.2 Constructor & Destructor Documentation . . . 438
8.249.2.1 MemoryRequest() [1/3] . . . 438
8.249.2.2 MemoryRequest() [2/3] . . . 439
8.249.2.3 MemoryRequest() [3/3] . . . 439
8.249.3 Member Data Documentation . . . 439
8.249.3.1 end . . . 439
8.249.3.2 loc . . . 439
8.249.3.3 offset . . . 440
8.249.3.4 size . . . 440
8.249.3.5 start . . . 440
8.250 tflite::MetadataBuilder Struct Reference . . . 440
8.250.1 Detailed Description . . . 441
8.251 tflite::MirrorPadOptionsBuilder Struct Reference . . . 441
8.251.1 Detailed Description . . . 441
8.252 tflite::ModelBuilder Struct Reference . . . 441
8.252.1 Detailed Description . . . 442
8.253 nntrainer::props::MoL_K Class Reference . . . 442
8.253.1 Detailed Description . . . 443
8.253.2 Member Typedef Documentation . . . 443
8.253.2.1 prop_tag . . . 443
8.253.3 Member Data Documentation . . . 443
8.253.3.1 key . . . 444
8.254 nntrainer::props::Momentum Class Reference . . . 444
8.254.1 Detailed Description . . . 445
8.254.2 Member Typedef Documentation . . . 445
8.254.2.1 prop_tag . . . 445
8.254.3 Constructor & Destructor Documentation . . . 445
8.254.3.1 Momentum() . . . 445
8.254.4 Member Function Documentation . . . 445
8.254.4.1 isValid() . . . 445
8.254.5 Member Data Documentation . . . 446
8.254.5.1 key . . . 446
8.255 tflite::MulOptionsBuilder Struct Reference . . . 446
8.255.1 Detailed Description . . . 447
8.256 nntrainer::MultioutRealizer Class Reference . . . 447
8.256.1 Detailed Description . . . 448
8.256.2 Constructor & Destructor Documentation . . . 448
8.256.2.1 ~MultioutRealizer() . . . 448
8.256.3 Member Function Documentation . . . 448
8.256.3.1 realize() . . . 448
8.257 nntrainer::props::Name Class Reference . . . 449
8.257.1 Detailed Description . . . 450

8.257.2 Member Typedef Documentation . . . 450
8.257.2.1 prop_tag . . . 450
8.257.3 Constructor & Destructor Documentation . . . 450
8.257.3.1 Name() [1/2] . . . 450
8.257.3.2 Name() [2/2] . . . 450
8.257.4 Member Function Documentation . . . 451
8.257.4.1 isValid() . . . 451
8.257.4.2 set() . . . 451
8.257.5 Member Data Documentation . . . 452
8.257.5.1 key . . . 452
8.258 tflite::NegOptionsBuilder Struct Reference . . . 452
8.258.1 Detailed Description . . . 453
8.259 tflite::NonMaxSuppressionV4OptionsBuilder Struct Reference . . . 453
8.259.1 Detailed Description . . . 453
8.260 tflite::NonMaxSuppressionV5OptionsBuilder Struct Reference . . . 453
8.260.1 Detailed Description . . . 454
8.261 nntrainer::props::Normalization Class Reference . . . 454
8.261.1 Detailed Description . . . 455
8.261.2 Constructor & Destructor Documentation . . . 455
8.261.2.1 Normalization() . . . 455
8.262 nntrainer::exception::not_supported Struct Reference . . . 456
8.262.1 Detailed Description . . . 456
8.263 tflite::NotEqualOptionsBuilder Struct Reference . . . 457
8.263.1 Detailed Description . . . 457
8.264 nntrainer::props::NumClass Class Reference . . . 457
8.264.1 Detailed Description . . . 458
8.264.2 Member Typedef Documentation . . . 458
8.264.2.1 prop_tag . . . 458
8.264.3 Member Function Documentation . . . 459
8.264.3.1 isValid() . . . 459
8.264.4 Member Data Documentation . . . 459
8.264.4.1 key . . . 459
8.265 nntrainer::props::NumHeads Class Reference . . . 459
8.265.1 Detailed Description . . . 460
8.265.2 Member Typedef Documentation . . . 460
8.265.2.1 prop_tag . . . 460
8.265.3 Constructor & Destructor Documentation . . . 461
8.265.3.1 NumHeads() . . . 461
8.265.4 Member Data Documentation . . . 461
8.265.4.1 key . . . 461
8.266 tflite::OneHotOptionsBuilder Struct Reference . . . 461
8.266.1 Detailed Description . . . 462

8.267 Op Class Reference . . . 462
8.267.1 Detailed Description . . . 462
8.268 tflite::OperatorBuilder Struct Reference . . . 462
8.268.1 Detailed Description . . . 463
8.269 tflite::OperatorCodeBuilder Struct Reference . . . 463
8.269.1 Detailed Description . . . 463
8.270 nntrainer::OptimizedV1Planner Class Reference . . . 464
8.270.1 Detailed Description . . . 465
8.270.2 Constructor & Destructor Documentation . . . 465
8.270.2.1 OptimizedV1Planner() . . . 465
8.270.3 Member Function Documentation . . . 465
8.270.3.1 getType() . . . 465
8.270.3.2 planLayout() . . . 466
8.271 nntrainer::OptimizedV2Planner Class Reference . . . 466
8.271.1 Detailed Description . . . 467
8.271.2 Constructor & Destructor Documentation . . . 468
8.271.2.1 OptimizedV2Planner() . . . 468
8.271.3 Member Function Documentation . . . 468
8.271.3.1 getType() . . . 468
8.271.3.2 planLayout() . . . 468
8.272 nntrainer::OptimizedV3Planner Class Reference . . . 469
8.272.1 Detailed Description . . . 470
8.272.2 Constructor & Destructor Documentation . . . 470
8.272.2.1 OptimizedV3Planner() . . . 470
8.272.3 Member Function Documentation . . . 470
8.272.3.1 getType() . . . 470
8.272.3.2 planLayout() . . . 471
8.273 nntrainer::props::OutDim Class Reference . . . 471
8.273.1 Detailed Description . . . 472
8.273.2 Member Typedef Documentation . . . 472
8.273.2.1 prop_tag . . . 473
8.273.3 Member Data Documentation . . . 473
8.273.3.1 key . . . 473
8.274 nntrainer::props::OutputLayer Class Reference . . . 473
8.274.1 Detailed Description . . . 474
8.274.2 Constructor & Destructor Documentation . . . 474
8.274.2.1 OutputLayer() [1/2] . . . 475
8.274.2.2 OutputLayer() [2/2] . . . 475
8.275 nntrainer::props::OutputShape Class Reference . . . 475
8.275.1 Detailed Description . . . 476
8.275.2 Member Typedef Documentation . . . 476
8.275.2.1 prop_tag . . . 476

8.275.3 Member Data Documentation . . . 476
8.275.3.1 key . . . 477
8.276 tflite::PackOptionsBuilder Struct Reference . . . 477
8.276.1 Detailed Description . . . 477
8.277 nntrainer::props::Padding1D Class Reference . . . 478
8.277.1 Detailed Description . . . 479
8.277.2 Member Typedef Documentation . . . 479
8.277.2.1 prop_tag . . . 479
8.277.3 Member Function Documentation . . . 479
8.277.3.1 compute() . . . 479
8.277.3.2 isValid() . . . 480
8.277.4 Member Data Documentation . . . 480
8.277.4.1 key . . . 480
8.278 nntrainer::props::Padding2D Class Reference . . . 481
8.278.1 Detailed Description . . . 482
8.278.2 Member Typedef Documentation . . . 482
8.278.2.1 prop_tag . . . 482
8.278.3 Member Function Documentation . . . 482
8.278.3.1 compute() . . . 482
8.278.3.2 isValid() . . . 483
8.278.4 Member Data Documentation . . . 483
8.278.4.1 key . . . 483
8.279 nntrainer::props::Padding_ Class Reference . . . 484
8.279.1 Detailed Description . . . 484
8.279.2 Member Typedef Documentation . . . 484
8.279.2.1 prop_tag . . . 485
8.280 tflite::PadOptionsBuilder Struct Reference . . . 485
8.280.1 Detailed Description . . . 485
8.281 tflite::PadV2OptionsBuilder Struct Reference . . . 485
8.281.1 Detailed Description . . . 486
8.282 nntrainer::ParallelBatch Class Reference . . . 486
8.282.1 Detailed Description . . . 486
8.282.2 Constructor & Destructor Documentation . . . 486
8.282.2.1 ParallelBatch() [1/2] . . . 486
8.282.2.2 ParallelBatch() [2/2] . . . 487
8.282.2.3 ∼ParallelBatch() . . . 487
8.282.3 Member Function Documentation . . . 487
8.282.3.1 getNumWorkers() . . . 487
8.282.3.2 run() . . . 488
8.282.3.3 setCallback() . . . 488
8.283 nntrainer::exception::permission_denied Struct Reference . . . 488
8.283.1 Detailed Description . . . 489

8.284 nntrainer::props::PermuteDims Class Reference . . . 489
8.284.1 Detailed Description . . . 490
8.284.2 Member Typedef Documentation . . . 490
8.284.2.1 prop_tag . . . 490
8.284.3 Member Function Documentation . . . 491
8.284.3.1 isValid() . . . 491
8.284.4 Member Data Documentation . . . 491
8.284.4.1 key . . . 491
8.285 nntrainer::PermuteLayer Class Reference . . . 492
8.285.1 Detailed Description . . . 493
8.285.2 Constructor & Destructor Documentation . . . 493
8.285.2.1 PermuteLayer() [1/2] . . . 493
8.285.2.2 PermuteLayer() [2/2] . . . 493
8.285.3 Member Function Documentation . . . 493
8.285.3.1 calcDerivative() . . . 494
8.285.3.2 exportTo() . . . 494
8.285.3.3 finalize() . . . 494
8.285.3.4 forwarding() . . . 495
8.285.3.5 getType() . . . 495
8.285.3.6 operator=() . . . 495
8.285.3.7 setProperty() . . . 495
8.285.3.8 supportBackwarding() . . . 496
8.286 nntrainer::internal::PluggedLayer Class Reference . . . 496
8.286.1 Detailed Description . . . 497
8.286.2 Constructor & Destructor Documentation . . . 497
8.286.2.1 PluggedLayer() [1/2] . . . 497
8.286.2.2 ∼PluggedLayer() . . . 498
8.286.2.3 PluggedLayer() [2/2] . . . 498
8.286.3 Member Function Documentation . . . 498
8.286.3.1 calcDerivative() . . . 498
8.286.3.2 calcGradient() . . . 498
8.286.3.3 exportTo() . . . 499
8.286.3.4 finalize() . . . 499
8.286.3.5 forwarding() . . . 499
8.286.3.6 getType() . . . 499
8.286.3.7 operator=() . . . 499
8.286.3.8 requireLabel() . . . 500
8.286.3.9 setBatch() . . . 500
8.286.3.10 setProperty() . . . 500
8.286.3.11 supportBackwarding() . . . 500
8.286.3.12 supportInPlace() . . . 500
8.287 nntrainer::internal::PluggedOptimizer Class Reference . . . 501

8.287.1 Detailed Description . . . 502
8.287.2 Constructor & Destructor Documentation . . . 502
8.287.2.1 PluggedOptimizer() [1/2] . . . 502
8.287.2.2 ~PluggedOptimizer() . . . 502
8.287.2.3 PluggedOptimizer() [2/2] . . . 503
8.287.3 Member Function Documentation . . . 503
8.287.3.1 applyGradient() . . . 503
8.287.3.2 getDefaultLearningRate() . . . 503
8.287.3.3 getOptimizerVariableDim() . . . 503
8.287.3.4 getType() . . . 504
8.287.3.5 operator=() . . . 504
8.287.3.6 read() . . . 504
8.287.3.7 save() . . . 505
8.287.3.8 setProperty() . . . 505
8.288 tflite::Pool2DOptionsBuilder Struct Reference . . . 506
8.288.1 Detailed Description . . . 506
8.289 nntrainer::props::PoolingType Class Reference . . . 506
8.289.1 Detailed Description . . . 507
8.290 nntrainer::props::PoolingTypeInfo Struct Reference . . . 507
8.290.1 Detailed Description . . . 508
8.290.2 Member Data Documentation . . . 508
8.290.2.1 EnumList . . . 508
8.290.2.2 EnumStr . . . 508
8.291 nntrainer::props::PoolSize Class Reference . . . 509
8.291.1 Detailed Description . . . 510
8.291.2 Member Typedef Documentation . . . 510
8.291.2.1 prop_tag . . . 510
8.291.3 Constructor & Destructor Documentation . . . 510
8.291.3.1 PoolSize() [1/2] . . . 510
8.291.3.2 PoolSize() [2/2] . . . 510
8.291.4 Member Data Documentation . . . 511
8.291.4.1 key . . . 511
8.292 tflite::PowOptionsBuilder Struct Reference . . . 511
8.292.1 Detailed Description . . . 511
8.293 nntrainer::PreprocessL2NormLayer Class Reference . . . 512
8.293.1 Detailed Description . . . 513
8.293.2 Constructor & Destructor Documentation . . . 513
8.293.2.1 PreprocessL2NormLayer() . . . 513
8.293.2.2 ~PreprocessL2NormLayer() . . . 513
8.293.3 Member Function Documentation . . . 513
8.293.3.1 calcDerivative() . . . 514
8.293.3.2 exportTo() . . . 514

8.293.3.3 finalize() . . . 514
8.293.3.4 forwarding() . . . 515
8.293.3.5 getType() . . . 515
8.293.3.6 setProperty() . . . 515
8.293.3.7 supportBackwarding() . . . 515
8.294 nntrainer::PreviousInputRealizer Class Reference . . . 516
8.294.1 Detailed Description . . . 517
8.294.2 Constructor & Destructor Documentation . . . 517
8.294.2.1 PreviousInputRealizer() . . . 517
8.294.2.2 ∼PreviousInputRealizer() . . . 517
8.294.3 Member Function Documentation . . . 517
8.294.3.1 realize() . . . 517
8.295 nntrainer::profile::ProfileEventData Struct Reference . . . 518
8.295.1 Detailed Description . . . 518
8.295.2 Constructor & Destructor Documentation . . . 518
8.295.2.1 ProfileEventData() [1/2] . . . 518
8.295.2.2 ProfileEventData() [2/2] . . . 519
8.296 nntrainer::profile::ProfileListener Class Reference . . . 519
8.296.1 Detailed Description . . . 520
8.296.2 Constructor & Destructor Documentation . . . 520
8.296.2.1 ProfileListener() . . . 520
8.296.2.2 ∼ProfileListener() . . . 520
8.296.3 Member Function Documentation . . . 520
8.296.3.1 notify() . . . 520
8.296.3.2 report() . . . 521
8.296.3.3 reset() . . . 521
8.296.3.4 result() . . . 521
8.297 nntrainer::profile::Profiler Class Reference . . . 522
8.297.1 Detailed Description . . . 522
8.297.2 Constructor & Destructor Documentation . . . 522
8.297.2.1 Profiler() [1/2] . . . 523
8.297.2.2 Profiler() [2/2] . . . 523
8.297.3 Member Function Documentation . . . 523
8.297.3.1 alloc() . . . 523
8.297.3.2 annotate() . . . 523
8.297.3.3 dealloc() . . . 524
8.297.3.4 end() . . . 524
8.297.3.5 Global() . . . 525
8.297.3.6 registerTimeItem() . . . 525
8.297.3.7 start() . . . 526
8.297.3.8 subscribe() . . . 526
8.297.3.9 unsubscribe() . . . 526

8.298 nntrainer::props::ProjectedKeyDim Class Reference . . . 527
8.298.1 Detailed Description . . . 528
8.298.2 Member Typedef Documentation . . . 528
8.298.2.1 prop_tag . . . 528
8.298.3 Member Data Documentation . . . 528
8.298.3.1 key . . . 528
8.299 nntrainer::props::ProjectedValueDim Class Reference . . . 529
8.299.1 Detailed Description . . . 530
8.299.2 Member Typedef Documentation . . . 530
8.299.2.1 prop_tag . . . 530
8.299.3 Member Data Documentation . . . 530
8.299.3.1 key . . . 530
8.300 nntrainer::PropsBufferSize Class Reference . . . 531
8.300.1 Detailed Description . . . 532
8.300.2 Member Typedef Documentation . . . 532
8.300.2.1 prop_tag . . . 532
8.300.3 Constructor & Destructor Documentation . . . 532
8.300.3.1 PropsBufferSize() . . . 532
8.300.4 Member Data Documentation . . . 532
8.300.4.1 key . . . 532
8.301 nntrainer::PropsMax Class Reference . . . 533
8.301.1 Detailed Description . . . 534
8.301.2 Member Typedef Documentation . . . 534
8.301.2.1 prop_tag . . . 534
8.301.3 Constructor & Destructor Documentation . . . 534
8.301.3.1 PropsMax() . . . 534
8.301.4 Member Data Documentation . . . 534
8.301.4.1 key . . . 534
8.302 nntrainer::PropsMin Class Reference . . . 535
8.302.1 Detailed Description . . . 536
8.302.2 Member Typedef Documentation . . . 536
8.302.2.1 prop_tag . . . 536
8.302.3 Constructor & Destructor Documentation . . . 536
8.302.3.1 PropsMin() . . . 536
8.302.4 Member Data Documentation . . . 536
8.302.4.1 key . . . 536
8.303 nntrainer::PropsNNSModelPath Class Reference . . . 537
8.303.1 Detailed Description . . . 538
8.303.2 Member Typedef Documentation . . . 538
8.303.2.1 prop_tag . . . 538
8.303.3 Member Function Documentation . . . 538
8.303.3.1 isValid() . . . 538

8.303.4 Member Data Documentation . . . 538
8.303.4.1 key . . . 539
8.304 nntrainer::PropsNumSamples Class Reference . . . 539
8.304.1 Detailed Description . . . 540
8.304.2 Member Typedef Documentation . . . 540
8.304.2.1 prop_tag . . . 540
8.304.3 Constructor & Destructor Documentation . . . 540
8.304.3.1 PropsNumSamples() . . . 540
8.304.4 Member Data Documentation . . . 541
8.304.4.1 key . . . 541
8.305 nntrainer::PropsTflModelPath Class Reference . . . 541
8.305.1 Detailed Description . . . 542
8.305.2 Member Typedef Documentation . . . 542
8.305.2.1 prop_tag . . . 542
8.305.3 Member Function Documentation . . . 543
8.305.3.1 isValid() . . . 543
8.305.4 Member Data Documentation . . . 543
8.305.4.1 key . . . 543
8.306 nntrainer::props::PropsUserData Class Reference . . . 544
8.306.1 Detailed Description . . . 545
8.307 tflite::QuantizationDetailsTraits< T > Struct Template Reference . . . 545
8.307.1 Detailed Description . . . 545
8.308 tflite::QuantizationDetailsTraits< tflite::CustomQuantization > Struct Reference . . . 545
8.308.1 Detailed Description . . . 545
8.309 tflite::QuantizationParametersBuilder Struct Reference . . . 546
8.309.1 Detailed Description . . . 546
8.310 tflite::QuantizeOptionsBuilder Struct Reference . . . 546
8.310.1 Detailed Description . . . 547
8.311 nntrainer::RandomDataOneHotProducer Class Reference . . . 547
8.311.1 Detailed Description . . . 548
8.311.2 Constructor & Destructor Documentation . . . 548
8.311.2.1 RandomDataOneHotProducer() . . . 548
8.311.2.2 ~RandomDataOneHotProducer() . . . 548
8.311.3 Member Function Documentation . . . 549
8.311.3.1 finalize() . . . 549
8.311.3.2 getType() . . . 549
8.311.3.3 isMultiThreadSafe() . . . 550
8.311.3.4 setProperty() . . . 550
8.311.3.5 size() . . . 551
8.312 nntrainer::props::RandomTranslate Class Reference . . . 551
8.312.1 Detailed Description . . . 552
8.312.2 Member Typedef Documentation . . . 552

8.312.2.1 prop_tag . . . 552
8.312.3 Member Function Documentation . . . 552
8.312.3.1 set() . . . 552
8.312.4 Member Data Documentation . . . 553
8.312.4.1 key . . . 553
8.313 tflite::RangeOptionsBuilder Struct Reference . . . 553
8.313.1 Detailed Description . . . 553
8.314 tflite::RankOptionsBuilder Struct Reference . . . 554
8.314.1 Detailed Description . . . 554
8.315 nntrainer::RawFileDataProducer Class Reference . . . 554
8.315.1 Detailed Description . . . 555
8.315.2 Constructor & Destructor Documentation . . . 555
8.315.2.1 RawFileDataProducer() [1/2] . . . 556
8.315.2.2 RawFileDataProducer() [2/2] . . . 556
8.315.2.3 ~RawFileDataProducer() . . . 556
8.315.3 Member Function Documentation . . . 556
8.315.3.1 exportTo() . . . 556
8.315.3.2 finalize() . . . 557
8.315.3.3 getType() . . . 557
8.315.3.4 setProperty() . . . 557
8.315.3.5 size() . . . 558
8.315.4 Member Data Documentation . . . 558
8.315.4.1 pixel_size . . . 558
8.316 nntrainer::props::RecurrentActivation Class Reference . . . 559
8.316.1 Detailed Description . . . 560
8.316.2 Constructor & Destructor Documentation . . . 560
8.316.2.1 RecurrentActivation() . . . 560
8.317 nntrainer::props::RecurrentInput Class Reference . . . 560
8.317.1 Detailed Description . . . 561
8.317.2 Constructor & Destructor Documentation . . . 561
8.317.2.1 RecurrentInput() [1/2] . . . 561
8.317.2.2 RecurrentInput() [2/2] . . . 562
8.318 nntrainer::props::RecurrentOutput Class Reference . . . 562
8.318.1 Detailed Description . . . 563
8.318.2 Constructor & Destructor Documentation . . . 563
8.318.2.1 RecurrentOutput() [1/2] . . . 563
8.318.2.2 RecurrentOutput() [2/2] . . . 563
8.319 nntrainer::RecurrentRealizer Class Reference . . . 564
8.319.1 Detailed Description . . . 565
8.319.2 Constructor & Destructor Documentation . . . 565
8.319.2.1 RecurrentRealizer() [1/2] . . . 565
8.319.2.2 RecurrentRealizer() [2/2] . . . 566

8.319.2.3 ~RecurrentRealizer() . . . 566
8.319.3 Member Function Documentation . . . 566
8.319.3.1 realize() . . . 566
8.320 nntrainer::props::ReduceDimension Class Reference . . . 568
8.320.1 Detailed Description . . . 569
8.321 tflite::ReducerOptionsBuilder Struct Reference . . . 569
8.321.1 Detailed Description . . . 569
8.322 nntrainer::props::RegularizerInfo Struct Reference . . . 569
8.322.1 Detailed Description . . . 570
8.322.2 Member Data Documentation . . . 570
8.322.2.1 EnumList . . . 570
8.323 nntrainer::RemapRealizer Class Reference . . . 571
8.323.1 Detailed Description . . . 572
8.323.2 Constructor & Destructor Documentation . . . 572
8.323.2.1 RemapRealizer() [1/2] . . . 572
8.323.2.2 RemapRealizer() [2/2] . . . 572
8.323.2.3 ~RemapRealizer() . . . 572
8.323.3 Member Function Documentation . . . 573
8.323.3.1 realize() . . . 573
8.324 nntrainer::props::ResetAfter Class Reference . . . 573
8.324.1 Detailed Description . . . 574
8.324.2 Member Typedef Documentation . . . 575
8.324.2.1 prop_tag . . . 575
8.324.3 Constructor & Destructor Documentation . . . 575
8.324.3.1 ResetAfter() . . . 575
8.324.4 Member Data Documentation . . . 575
8.324.4.1 key . . . 575
8.325 tflite::ReshapeOptionsBuilder Struct Reference . . . 575
8.325.1 Detailed Description . . . 576
8.326 tflite::ResizeBilinearOptionsBuilder Struct Reference . . . 576
8.326.1 Detailed Description . . . 576
8.327 tflite::ResizeNearestNeighborOptionsBuilder Struct Reference . . . 577
8.327.1 Detailed Description . . . 577
8.328 nntrainer::props::ReturnAttentionWeight Class Reference . . . 577
8.328.1 Detailed Description . . . 578
8.328.2 Member Typedef Documentation . . . 578
8.328.2.1 prop_tag . . . 579
8.328.3 Constructor & Destructor Documentation . . . 579
8.328.3.1 ReturnAttentionWeight() . . . 579
8.328.4 Member Data Documentation . . . 579
8.328.4.1 key . . . 579
8.329 nntrainer::props::ReturnAttentionWeightInfo Struct Reference . . . 579

8.329.1 Detailed Description . . . 580
8.329.2 Member Data Documentation . . . 580
8.329.2.1 EnumList . . . 580
8.330 nntrainer::props::ReturnSequences Class Reference . . . 581
8.330.1 Detailed Description . . . 582
8.330.2 Constructor & Destructor Documentation . . . 582
8.330.2.1 ReturnSequences() . . . 582
8.331 tflite::ReverseSequenceOptionsBuilder Struct Reference . . . 582
8.331.1 Detailed Description . . . 583
8.332 tflite::ReverseV2OptionsBuilder Struct Reference . . . 583
8.332.1 Detailed Description . . . 583
8.333 tflite::RNNOptionsBuilder Struct Reference . . . 583
8.333.1 Detailed Description . . . 584
8.334 nntrainer::RunLayerContext Class Reference . . . 584
8.334.1 Detailed Description . . . 586
8.334.2 Constructor & Destructor Documentation . . . 586
8.334.2.1 RunLayerContext() [1/3] . . . 586
8.334.2.2 RunLayerContext() [2/3] . . . 586
8.334.2.3 RunLayerContext() [3/3] . . . 586
8.334.3 Member Function Documentation . . . 587
8.334.3.1 executeInPlace() . . . 587
8.334.3.2 getIncomingDerivative() . . . 587
8.334.3.3 getInput() [1/2] . . . 588
8.334.3.4 getInput() [2/2] . . . 589
8.334.3.5 getInputGrad() . . . 589
8.334.3.6 getLabel() . . . 590
8.334.3.7 getLoss() . . . 591
8.334.3.8 getName() . . . 591
8.334.3.9 getNumInputs() . . . 591
8.334.3.10 getNumOutputs() . . . 592
8.334.3.11 getNumTensors() . . . 592
8.334.3.12 getNumWeightOptVar() . . . 592
8.334.3.13 getNumWeights() . . . 593
8.334.3.14 getOutgoingDerivative() . . . 593
8.334.3.15 getOutput() [1/2] . . . 594
8.334.3.16 getOutput() [2/2] . . . 595
8.334.3.17 getOutputGrad() . . . 595
8.334.3.18 getOutputGradUnsafe() . . . 596
8.334.3.19 getRegularizationLoss() . . . 597
8.334.3.20 getTensor() [1/2] . . . 598
8.334.3.21 getTensor() [2/2] . . . 598
8.334.3.22 getTensorGrad() [1/2] . . . 599

8.334.3.23 getTensorGrad() [2/2] . . . 599
8.334.3.24 getTensorName() . . . 599
8.334.3.25 getTrainable() . . . 600
8.334.3.26 getWeight() . . . 600
8.334.3.27 getWeightGrad() . . . 601
8.334.3.28 getWeightName() . . . 601
8.334.3.29 getWeightObject() . . . 602
8.334.3.30 getWeightOptVar() . . . 602
8.334.3.31 inputHasGradient() . . . 603
8.334.3.32 isGradientClipByGlobalNorm() . . . 603
8.334.3.33 isGradientFirstAccess() . . . 604
8.334.3.34 isGradientLastAccess() . . . 604
8.334.3.35 isLabelAvailable() . . . 605
8.334.3.36 isWeightDependent() . . . 605
8.334.3.37 outputHasGradient() . . . 606
8.334.3.38 readyToUse() . . . 606
8.334.3.39 setBatch() . . . 607
8.334.3.40 setLoss() . . . 607
8.334.3.41 tensorHasGradient() . . . 608
8.334.3.42 updateTensor() . . . 608
8.334.3.43 validate() . . . 608
8.334.3.44 weightHasGradient() . . . 609
8.335 nntrainer::RunOptimizerContext Class Reference . . . 610
8.335.1 Detailed Description . . . 610
8.335.2 Constructor & Destructor Documentation . . . 610
8.335.2.1 RunOptimizerContext() . . . 610
8.335.3 Member Function Documentation . . . 610
8.335.3.1 applyGradient() . . . 610
8.335.3.2 getGradient() . . . 611
8.335.3.3 getIteration() . . . 612
8.335.3.4 getLearningRate() . . . 612
8.335.3.5 getOptimizerVariable() . . . 612
8.335.3.6 getWeight() . . . 613
8.335.3.7 readyToUse() . . . 613
8.336 nntrainer::Sample Class Reference . . . 614
8.336.1 Detailed Description . . . 614
8.336.2 Constructor & Destructor Documentation . . . 614
8.336.2.1 Sample() . . . 614
8.336.3 Member Function Documentation . . . 615
8.336.3.1 getInputsRef() [1/2] . . . 615
8.336.3.2 getInputsRef() [2/2] . . . 615
8.336.3.3 getLabelsRef() [1/2] . . . 615

8.336.3.4 getLabelsRef() [2/2] . . . 616
8.337 tflite::ScatterNdOptionsBuilder Struct Reference . . . 616
8.337.1 Detailed Description . . . 616
8.338 nntrainer::ScopedView< T > Class Template Reference . . . 616
8.338.1 Detailed Description . . . 617
8.338.2 Constructor & Destructor Documentation . . . 617
8.338.2.1 ScopedView() . . . 617
8.338.2.2 ~ScopedView() . . . 618
8.338.3 Member Function Documentation . . . 618
8.338.3.1 get() [1/2] . . . 618
8.338.3.2 get() [2/2] . . . 618
8.338.3.3 isEmpty() . . . 619
8.339 tflite::SegmentSumOptionsBuilder Struct Reference . . . 619
8.339.1 Detailed Description . . . 619
8.340 tflite::SelectOptionsBuilder Struct Reference . . . 619
8.340.1 Detailed Description . . . 620
8.341 tflite::SelectV2OptionsBuilder Struct Reference . . . 620
8.341.1 Detailed Description . . . 620
8.342 tflite::SequenceRNNOptionsBuilder Struct Reference . . . 621
8.342.1 Detailed Description . . . 621
8.343 tflite::ShapeOptionsBuilder Struct Reference . . . 621
8.343.1 Detailed Description . . . 622
8.344 nntrainer::props::SharedFrom Class Reference . . . 622
8.344.1 Detailed Description . . . 623
8.344.2 Member Typedef Documentation . . . 623
8.344.2.1 prop_tag . . . 623
8.344.3 Member Data Documentation . . . 623
8.344.3.1 key . . . 623
8.345 tflite::SignatureDefBuilder Struct Reference . . . 624
8.345.1 Detailed Description . . . 624
8.346 tflite::SkipGramOptionsBuilder Struct Reference . . . 624
8.346.1 Detailed Description . . . 625
8.347 tflite::SliceOptionsBuilder Struct Reference . . . 625
8.347.1 Detailed Description . . . 625
8.348 nntrainer::SliceRealizer Class Reference . . . 626
8.348.1 Detailed Description . . . 626
8.348.2 Constructor & Destructor Documentation . . . 627
8.348.2.1 SliceRealizer() . . . 627
8.348.2.2 ~SliceRealizer() . . . 627
8.348.3 Member Function Documentation . . . 627
8.348.3.1 realize() . . . 627
8.349 tflite::SoftmaxOptionsBuilder Struct Reference . . . 628

8.349.1 Detailed Description . . . 629
8.350 tflite::SpaceToBatchNDOptionsBuilder Struct Reference . . . 629
8.350.1 Detailed Description . . . 629
8.351 tflite::SpaceToDepthOptionsBuilder Struct Reference . . . 630
8.351.1 Detailed Description . . . 630
8.352 tflite::SparseIndexVectorTraits< T > Struct Template Reference . . . 630
8.352.1 Detailed Description . . . 630
8.353 tflite::SparseIndexVectorTraits< tflite::Int32Vector > Struct Reference . . . 631
8.353.1 Detailed Description . . . 631
8.354 tflite::SparseIndexVectorTraits< tflite::Uint16Vector > Struct Reference . . . 631
8.354.1 Detailed Description . . . 631
8.355 tflite::SparseIndexVectorTraits< tflite::Uint8Vector > Struct Reference . . . 631
8.355.1 Detailed Description . . . 631
8.356 tflite::SparseToDenseOptionsBuilder Struct Reference . . . 632
8.356.1 Detailed Description . . . 632
8.357 tflite::SparsityParametersBuilder Struct Reference . . . 632
8.357.1 Detailed Description . . . 633
8.358 nntrainer::props::SplitDimension Class Reference . . . 633
8.358.1 Detailed Description . . . 634
8.358.2 Member Function Documentation . . . 634
8.358.2.1 isValid() . . . 634
8.359 nntrainer::props::SplitNumber Class Reference . . . 635
8.359.1 Detailed Description . . . 636
8.359.2 Member Typedef Documentation . . . 636
8.359.2.1 prop_tag . . . 636
8.359.3 Member Data Documentation . . . 636
8.359.3.1 key . . . 636
8.360 tflite::SplitOptionsBuilder Struct Reference . . . 637
8.360.1 Detailed Description . . . 637
8.361 tflite::SplitVOptionsBuilder Struct Reference . . . 637
8.361.1 Detailed Description . . . 638
8.362 tflite::SquaredDifferenceOptionsBuilder Struct Reference . . . 638
8.362.1 Detailed Description . . . 638
8.363 tflite::SquareOptionsBuilder Struct Reference . . . 638
8.363.1 Detailed Description . . . 639
8.364 tflite::SqueezeOptionsBuilder Struct Reference . . . 639
8.364.1 Detailed Description . . . 639
8.365 nntrainer::SrcSharedTensor Class Reference . . . 639
8.365.1 Detailed Description . . . 640
8.366 nntrainer::props::Standardization Class Reference . . . 640
8.366.1 Detailed Description . . . 641
8.366.2 Constructor & Destructor Documentation . . . 641

8.366.2.1 Standardization() . . . 641
8.367 nntrainer::props::Stride Class Reference . . . 642
8.367.1 Detailed Description . . . 643
8.367.2 Member Typedef Documentation . . . 643
8.367.2.1 prop_tag . . . 643
8.367.3 Constructor & Destructor Documentation . . . 643
8.367.3.1 Stride() . . . 643
8.367.4 Member Data Documentation . . . 643
8.367.4.1 key . . . 643
8.368 tflite::StridedSliceOptionsBuilder Struct Reference . . . 644
8.368.1 Detailed Description . . . 644
8.369 tflite::SubGraphBuilder Struct Reference . . . 644
8.369.1 Detailed Description . . . 645
8.370 tflite::SubOptionsBuilder Struct Reference . . . 645
8.370.1 Detailed Description . . . 645
8.371 tflite::SVDFOptionsBuilder Struct Reference . . . 645
8.371.1 Detailed Description . . . 646
8.372 nntrainer::SwapDevice Class Reference . . . 646
8.372.1 Detailed Description . . . 647
8.372.2 Constructor & Destructor Documentation . . . 647
8.372.2.1 SwapDevice() [1/2] . . . 647
8.372.2.2 SwapDevice() [2/2] . . . 647
8.372.2.3 ∼SwapDevice() . . . 647
8.372.3 Member Function Documentation . . . 647
8.372.3.1 finish() . . . 648
8.372.3.2 getBuffer() . . . 648
8.372.3.3 getDevicePath() . . . 648
8.372.3.4 isOperating() . . . 649
8.372.3.5 putBuffer() . . . 649
8.372.3.6 start() . . . 649
8.372.4 Member Data Documentation . . . 649
8.372.4.1 swap_device_default_path . . . 650
8.373 nntrainer::props::TargetShape Class Reference . . . 650
8.373.1 Detailed Description . . . 651
8.373.2 Member Typedef Documentation . . . 651
8.373.2.1 prop_tag . . . 651
8.373.3 Member Data Documentation . . . 652
8.373.3.1 key . . . 652
8.374 nntrainer::Task Class Reference . . . 652
8.374.1 Detailed Description . . . 653
8.374.2 Constructor & Destructor Documentation . . . 653
8.374.2.1 Task() . . . 653

8.374.2.2 ~Task() . . . 653
8.374.3 Member Function Documentation . . . 654
8.374.3.1 done() . . . 654
8.374.3.2 getData() . . . 654
8.374.3.3 getType() . . . 654
8.374.3.4 getWork() . . . 655
8.374.3.5 setState() . . . 655
8.374.3.6 started() . . . 655
8.375 nntrainer::TaskAsync< T > Class Template Reference . . . 656
8.375.1 Detailed Description . . . 657
8.375.2 Constructor & Destructor Documentation . . . 657
8.375.2.1 TaskAsync() . . . 657
8.375.2.2 ~TaskAsync() . . . 657
8.375.3 Member Function Documentation . . . 657
8.375.3.1 getPriority() . . . 658
8.375.3.2 getTimeout() . . . 658
8.375.3.3 getType() . . . 658
8.375.3.4 setPriority() . . . 658
8.375.3.5 setTimeout() . . . 659
8.376 nntrainer::TaskExecutor Class Reference . . . 659
8.376.1 Detailed Description . . . 660
8.376.2 Member Typedef Documentation . . . 661
8.376.2.1 CompleteCallback . . . 661
8.376.2.2 TaskInfo . . . 661
8.376.3 Constructor & Destructor Documentation . . . 661
8.376.3.1 TaskExecutor() . . . 661
8.376.3.2 ~TaskExecutor() . . . 662
8.376.4 Member Function Documentation . . . 662
8.376.4.1 cancel() . . . 662
8.376.4.2 clean() . . . 662
8.376.4.3 handleWork() . . . 663
8.376.4.4 run() [1/2] . . . 663
8.376.4.5 run() [2/2] . . . 664
8.376.4.6 worker() . . . 664
8.377 tflite::TensorBuilder Struct Reference . . . 665
8.377.1 Detailed Description . . . 666
8.378 tflite::TensorMapBuilder Struct Reference . . . 666
8.378.1 Detailed Description . . . 666
8.379 nntrainer::TensorSpecV2 Struct Reference . . . 666
8.379.1 Detailed Description . . . 667
8.379.2 Member Enumeration Documentation . . . 667
8.379.2.1 RequestType . . . 667

8.379.3 Member Data Documentation . . . 668
8.379.3.1 additional_exec_order . . . 668
8.379.3.2 dim . . . 668
8.379.3.3 initializer . . . 668
8.379.3.4 ls . . . 668
8.379.3.5 name . . . 669
8.379.3.6 offset . . . 669
8.379.3.7 reference_name . . . 669
8.379.3.8 request_type . . . 669
8.380 nntrainer::TfliteInterpreter Class Reference . . . 670
8.380.1 Detailed Description . . . 670
8.380.2 Constructor & Destructor Documentation . . . 671
8.380.2.1 TfliteInterpreter() . . . 671
8.380.2.2 ∼TfliteInterpreter() . . . 671
8.380.3 Member Function Documentation . . . 671
8.380.3.1 deserialize() . . . 671
8.380.3.2 serialize() . . . 672
8.381 nntrainer::TfOpNode Class Reference . . . 673
8.381.1 Detailed Description . . . 674
8.381.2 Constructor & Destructor Documentation . . . 674
8.381.2.1 TfOpNode() . . . 674
8.381.3 Member Function Documentation . . . 674
8.381.3.1 arg() . . . 675
8.381.3.2 arity() [1/2] . . . 675
8.381.3.3 arity() [2/2] . . . 675
8.381.3.4 finalize() . . . 675
8.381.3.5 getBuiltinOps() . . . 676
8.381.3.6 getInputNodes() . . . 676
8.381.3.7 getInputs() [1/2] . . . 676
8.381.3.8 getInputs() [2/2] . . . 677
8.381.3.9 getOptionType() . . . 677
8.381.3.10 getOpType() . . . 677
8.381.3.11 getOutputs() [1/2] . . . 677
8.381.3.12 getOutputs() [2/2] . . . 678
8.381.3.13 getWeights() [1/2] . . . 678
8.381.3.14 getWeights() [2/2] . . . 678
8.381.3.15 isInputNode() . . . 678
8.381.3.16 isNeedReorder() . . . 679
8.381.3.17 isOutputNode() . . . 679
8.381.3.18 isTrainable() . . . 679
8.381.3.19 isVirtualNode() . . . 680
8.381.3.20 setArg() . . . 680

8.381.3.21 setBuiltinOptions() . . . 680
8.381.3.22 setInputTransformFn() . . . 681
8.381.3.23 setLayerNode() . . . 681
8.381.3.24 setNeedReorderWeight() . . . 682
8.381.3.25 setOpType() . . . 682
8.381.3.26 setTrainable() . . . 683
8.381.3.27 setWeightTransformFn() . . . 683
8.381.3.28 weightReorder() . . . 684
8.382 tflite::TileOptionsBuilder Struct Reference . . . 684
8.382.1 Detailed Description . . . 685
8.383 nntrainer::props::Timestep Class Reference . . . 685
8.383.1 Detailed Description . . . 686
8.383.2 Member Typedef Documentation . . . 686
8.383.2.1 prop_tag . . . 686
8.383.3 Member Data Documentation . . . 686
8.383.3.1 key . . . 686
8.384 tflite::TopKV2OptionsBuilder Struct Reference . . . 686
8.384.1 Detailed Description . . . 687
8.385 nntrainer::props::Trainable Class Reference . . . 687
8.385.1 Detailed Description . . . 688
8.385.2 Constructor & Destructor Documentation . . . 688
8.385.2.1 Trainable() . . . 688
8.386 tflite::TransposeConvOptionsBuilder Struct Reference . . . 689
8.386.1 Detailed Description . . . 689
8.387 tflite::TransposeOptionsBuilder Struct Reference . . . 689
8.387.1 Detailed Description . . . 690
8.388 tflite::Uint16VectorBuilder Struct Reference . . . 690
8.388.1 Detailed Description . . . 690
8.389 tflite::Uint8VectorBuilder Struct Reference . . . 690
8.389.1 Detailed Description . . . 691
8.390 tflite::UnidirectionalSequenceLSTMOptionsBuilder Struct Reference . . . 691
8.390.1 Detailed Description . . . 691
8.391 tflite::UniqueOptionsBuilder Struct Reference . . . 692
8.391.1 Detailed Description . . . 692
8.392 nntrainer::props::Unit Class Reference . . . 692
8.392.1 Detailed Description . . . 693
8.392.2 Member Typedef Documentation . . . 693
8.392.2.1 prop_tag . . . 693
8.392.3 Member Data Documentation . . . 693
8.392.3.1 key . . . 694
8.393 tflite::UnpackOptionsBuilder Struct Reference . . . 694
8.393.1 Detailed Description . . . 694

8.394 nntrainer::props::UnrollFor Class Reference . . . 695
8.394.1 Detailed Description . . . 696
8.395 nntrainer::Var_Grad Class Reference . . . 696
8.395.1 Detailed Description . . . 698
8.395.2 Member Typedef Documentation . . . 698
8.395.2.1 Spec . . . 698
8.395.3 Constructor & Destructor Documentation . . . 698
8.395.3.1 Var_Grad() [1/7] . . . 698
8.395.3.2 Var_Grad() [2/7] . . . 698
8.395.3.3 Var_Grad() [3/7] . . . 699
8.395.3.4 Var_Grad() [4/7] . . . 699
8.395.3.5 Var_Grad() [5/7] . . . 700
8.395.3.6 Var_Grad() [6/7] . . . 700
8.395.3.7 Var_Grad() [7/7] . . . 700
8.395.4 Member Function Documentation . . . 701
8.395.4.1 getDim() . . . 701
8.395.4.2 getGradient() . . . 701
8.395.4.3 getGradientName() . . . 701
8.395.4.4 getGradientNorm() . . . 702
8.395.4.5 getGradientRef() [1/2] . . . 702
8.395.4.6 getGradientRef() [2/2] . . . 702
8.395.4.7 getName() . . . 703
8.395.4.8 getVariable() . . . 703
8.395.4.9 getVariableRef() [1/2] . . . 704
8.395.4.10 getVariableRef() [2/2] . . . 704
8.395.4.11 hasGradient() . . . 704
8.395.4.12 initializeGradient() . . . 705
8.395.4.13 initializeVariable() . . . 705
8.395.4.14 isDependent() . . . 706
8.395.4.15 isGradientFirstAccess() . . . 706
8.395.4.16 isGradientLastAccess() . . . 706
8.395.4.17 operator=() [1/2] . . . 706
8.395.4.18 operator=() [2/2] . . . 707
8.395.4.19 resetGradient() . . . 707
8.395.4.20 setAsGradientFirstAccess() . . . 707
8.395.4.21 setAsGradientLastAccess() . . . 708
8.395.4.22 setBatchSize() . . . 708
8.395.5 Member Data Documentation . . . 708
8.395.5.1 grad . . . 708
8.395.5.2 is_dependent . . . 708
8.395.5.3 is_first_access_gradient . . . 709
8.395.5.4 is_last_access_gradient . . . 709

8.395.5.5 var . . . 709
8.396 nntrainer::VarGradSpecV2 Struct Reference . . . 709
8.396.1 Detailed Description . . . 710
8.396.2 Constructor & Destructor Documentation . . . 710
8.396.2.1 VarGradSpecV2() [1/3] . . . 710
8.396.2.2 VarGradSpecV2() [2/3] . . . 710
8.396.2.3 VarGradSpecV2() [3/3] . . . 711
8.396.3 Member Function Documentation . . . 711
8.396.3.1 operator=() . . . 711
8.396.4 Member Data Documentation . . . 711
8.396.4.1 gradient_spec . . . 711
8.396.4.2 variable_spec . . . 712
8.397 nntrainer::ViewQueue< T > Class Template Reference . . . 712
8.397.1 Detailed Description . . . 712
8.397.2 Member Function Documentation . . . 713
8.397.2.1 isEmpty() . . . 713
8.397.2.2 push() . . . 713
8.397.2.3 size() . . . 713
8.397.2.4 waitAndPop() . . . 714
8.398 nntrainer::Weight Class Reference . . . 715
8.398.1 Detailed Description . . . 717
8.398.2 Member Typedef Documentation . . . 717
8.398.2.1 Spec . . . 717
8.398.3 Constructor & Destructor Documentation . . . 717
8.398.3.1 Weight() [1/6] . . . 717
8.398.3.2 Weight() [2/6] . . . 718
8.398.3.3 Weight() [3/6] . . . 718
8.398.3.4 Weight() [4/6] . . . 719
8.398.3.5 Weight() [5/6] . . . 719
8.398.3.6 Weight() [6/6] . . . 719
8.398.4 Member Function Documentation . . . 720
8.398.4.1 clipGradientByGlobalNorm() . . . 720
8.398.4.2 clone() . . . 720
8.398.4.3 getNumOptVariable() . . . 720
8.398.4.4 getOptimizerVariableRef() . . . 721
8.398.4.5 isGradientClipByGlobalNorm() [1/2] . . . 721
8.398.4.6 isGradientClipByGlobalNorm() [2/2] . . . 721
8.398.4.7 isWeightDecay() . . . 722
8.398.4.8 isWeightRegularizerL2Norm() . . . 722
8.398.4.9 operator=() [1/2] . . . 723
8.398.4.10 operator=() [2/2] . . . 723
8.398.4.11 setOptimizerVariables() . . . 724

8.398.5 Friends And Related Function Documentation . . . 724
8.398.5.1 swap . . . 724
8.399 nntrainer::props::WeightDecay Class Reference . . . 725
8.399.1 Detailed Description . . . 726
8.399.2 Constructor & Destructor Documentation . . . 726
8.399.2.1 WeightDecay() . . . 726
8.399.3 Member Data Documentation . . . 726
8.399.3.1 key . . . 726
8.400 nntrainer::props::WeightInitializer Class Reference . . . 727
8.400.1 Detailed Description . . . 728
8.401 nntrainer::props::WeightRegularizer Class Reference . . . 728
8.401.1 Detailed Description . . . 729
8.402 nntrainer::props::WeightRegularizerConstant Class Reference . . . 730
8.402.1 Detailed Description . . . 731
8.402.2 Constructor & Destructor Documentation . . . 731
8.402.2.1 WeightRegularizerConstant() . . . 731
8.402.3 Member Data Documentation . . . 731
8.402.3.1 key . . . 731
8.403 nntrainer::WeightSpecV2 Struct Reference . . . 732
8.403.1 Detailed Description . . . 732
8.403.2 Member Data Documentation . . . 732
8.403.2.1 clip_by_global_norm . . . 733
8.403.2.2 decay . . . 733
8.403.2.3 regularizer . . . 733
8.403.2.4 regularizer_constant . . . 733
8.403.2.5 vg_spec . . . 733
8.404 nntrainer::WGradMemoryRequest Struct Reference . . . 734
8.404.1 Detailed Description . . . 734
8.405 tflite::WhereOptionsBuilder Struct Reference . . . 734
8.405.1 Detailed Description . . . 735
8.406 tflite::WhileOptionsBuilder Struct Reference . . . 735
8.406.1 Detailed Description . . . 735
8.407 nntrainer::props::ZeroIdxMask Class Reference . . . 736
8.407.1 Detailed Description . . . 736
8.407.2 Member Typedef Documentation . . . 737
8.407.2.1 prop_tag . . . 737
8.407.3 Member Data Documentation . . . 737
8.407.3.1 key . . . 737
8.408 tflite::ZerosLikeOptionsBuilder Struct Reference . . . 737
8.408.1 Detailed Description . . . 737

9 File Documentation **739**

9.1 app_context.cpp File Reference . . . 739
9.1.1 Detailed Description . . . 741
9.2 app_context.h File Reference . . . 741
9.2.1 Detailed Description . . . 743
9.3 compiler/activation_realizer.cpp File Reference . . . 743
9.3.1 Detailed Description . . . 744
9.4 compiler/activation_realizer.h File Reference . . . 744
9.4.1 Detailed Description . . . 745
9.5 compiler/bn_realizer.cpp File Reference . . . 746
9.5.1 Detailed Description . . . 746
9.6 compiler/bn_realizer.h File Reference . . . 747
9.6.1 Detailed Description . . . 748
9.7 compiler/compiler.h File Reference . . . 748
9.7.1 Detailed Description . . . 749
9.8 compiler/compiler_fwd.h File Reference . . . 749
9.8.1 Detailed Description . . . 750
9.9 compiler/flatbuffer_interpreter.cpp File Reference . . . 750
9.9.1 Detailed Description . . . 751
9.10 compiler/flatbuffer_interpreter.h File Reference . . . 751
9.10.1 Detailed Description . . . 752
9.11 compiler/flatbuffer_opnode.h File Reference . . . 752
9.11.1 Detailed Description . . . 753
9.12 compiler/flatten_realizer.cpp File Reference . . . 754
9.12.1 Detailed Description . . . 754
9.13 compiler/flatten_realizer.h File Reference . . . 755
9.13.1 Detailed Description . . . 756
9.14 compiler/ini_interpreter.cpp File Reference . . . 756
9.14.1 Detailed Description . . . 757
9.15 compiler/ini_interpreter.h File Reference . . . 757
9.15.1 Detailed Description . . . 758
9.16 compiler/input_realizer.cpp File Reference . . . 759
9.16.1 Detailed Description . . . 759
9.17 compiler/input_realizer.h File Reference . . . 760
9.17.1 Detailed Description . . . 761
9.18 compiler/interpreter.h File Reference . . . 761
9.18.1 Detailed Description . . . 762
9.19 compiler/loss_realizer.h File Reference . . . 763
9.19.1 Detailed Description . . . 764
9.20 compiler/multiout_realizer.h File Reference . . . 765
9.20.1 Detailed Description . . . 766
9.21 compiler/previous_input_realizer.cpp File Reference . . . 767
9.21.1 Detailed Description . . . 767

9.22 compiler/previous_input_realizer.h File Reference . . . 768
9.22.1 Detailed Description . . . 769
9.23 compiler/realizer.h File Reference . . . 769
9.23.1 Detailed Description . . . 770
9.24 compiler/recurrent_realizer.h File Reference . . . 770
9.24.1 Detailed Description . . . 771
9.25 compiler/remap_realizer.h File Reference . . . 772
9.25.1 Detailed Description . . . 773
9.26 compiler/slice_realizer.cpp File Reference . . . 774
9.26.1 Detailed Description . . . 774
9.27 compiler/slice_realizer.h File Reference . . . 775
9.27.1 Detailed Description . . . 776
9.28 compiler/tflite_interpreter.cpp File Reference . . . 776
9.28.1 Detailed Description . . . 777
9.29 compiler/tflite_interpreter.h File Reference . . . 777
9.29.1 Detailed Description . . . 778
9.30 compiler/tflite_opnode.cpp File Reference . . . 778
9.30.1 Detailed Description . . . 779
9.31 compiler/tflite_opnode.h File Reference . . . 779
9.31.1 Detailed Description . . . 780
9.32 dataset/data_iteration.cpp File Reference . . . 781
9.32.1 Detailed Description . . . 781
9.33 dataset/data_iteration.h File Reference . . . 782
9.33.1 Detailed Description . . . 783
9.34 dataset/data_producer.h File Reference . . . 783
9.34.1 Detailed Description . . . 784
9.35 dataset/databuffer.cpp File Reference . . . 784
9.35.1 Detailed Description . . . 785
9.36 dataset/databuffer.h File Reference . . . 785
9.36.1 Detailed Description . . . 786
9.37 dataset/databuffer_factory.cpp File Reference . . . 786
9.37.1 Detailed Description . . . 787
9.38 dataset/databuffer_factory.h File Reference . . . 787
9.38.1 Detailed Description . . . 788
9.39 dataset/dir_data_producers.h File Reference . . . 788
9.39.1 Detailed Description . . . 789
9.40 dataset/func_data_producer.cpp File Reference . . . 790
9.40.1 Detailed Description . . . 790
9.41 dataset/func_data_producer.h File Reference . . . 791
9.41.1 Detailed Description . . . 792
9.42 dataset/iteration_queue.cpp File Reference . . . 792
9.42.1 Detailed Description . . . 793

9.43 dataset/iteration_queue.h File Reference . . . 793
9.43.1 Detailed Description . . . 794
9.44 dataset/random_data_producers.cpp File Reference . . . 795
9.44.1 Detailed Description . . . 795
9.45 dataset/random_data_producers.h File Reference . . . 796
9.45.1 Detailed Description . . . 797
9.46 dataset/raw_file_data_producer.cpp File Reference . . . 797
9.46.1 Detailed Description . . . 798
9.47 dataset/raw_file_data_producer.h File Reference . . . 798
9.47.1 Detailed Description . . . 799
9.48 delegate.h File Reference . . . 800
9.48.1 Detailed Description . . . 800
9.49 graph/connection.cpp File Reference . . . 801
9.49.1 Detailed Description . . . 801
9.50 graph/connection.h File Reference . . . 802
9.50.1 Detailed Description . . . 802
9.51 graph/graph_node.h File Reference . . . 803
9.51.1 Detailed Description . . . 804
9.52 graph/network_graph.h File Reference . . . 804
9.52.1 Detailed Description . . . 804
9.53 layers/acti_func.cpp File Reference . . . 806
9.53.1 Detailed Description . . . 806
9.54 layers/activation_layer.cpp File Reference . . . 807
9.54.1 Detailed Description . . . 808
9.55 layers/activation_layer.h File Reference . . . 808
9.55.1 Detailed Description . . . 809
9.56 layers/addition_layer.cpp File Reference . . . 809
9.56.1 Detailed Description . . . 810
9.57 layers/addition_layer.h File Reference . . . 810
9.57.1 Detailed Description . . . 810
9.58 layers/attention_layer.cpp File Reference . . . 811
9.58.1 Detailed Description . . . 811
9.59 layers/attention_layer.h File Reference . . . 812
9.59.1 Detailed Description . . . 812
9.60 layers/bn_layer.cpp File Reference . . . 813
9.60.1 Detailed Description . . . 813
9.61 layers/bn_layer.h File Reference . . . 814
9.61.1 Detailed Description . . . 814
9.62 layers/centroid_knn.cpp File Reference . . . 815
9.62.1 Detailed Description . . . 815
9.63 layers/centroid_knn.h File Reference . . . 816
9.63.1 Detailed Description . . . 816

9.64 layers/common_properties.cpp File Reference . . . 817
9.64.1 Detailed Description . . . 818
9.65 layers/common_properties.h File Reference . . . 818
9.65.1 Detailed Description . . . 822
9.66 layers/concat_layer.cpp File Reference . . . 823
9.66.1 Detailed Description . . . 823
9.67 layers/concat_layer.h File Reference . . . 824
9.67.1 Detailed Description . . . 824
9.68 layers/conv1d_layer.h File Reference . . . 824
9.68.1 Detailed Description . . . 825
9.69 layers/conv2d_layer.h File Reference . . . 825
9.69.1 Detailed Description . . . 826
9.70 layers/dropout.h File Reference . . . 826
9.70.1 Detailed Description . . . 827
9.71 layers/embedding.cpp File Reference . . . 827
9.71.1 Detailed Description . . . 828
9.72 layers/embedding.h File Reference . . . 828
9.72.1 Detailed Description . . . 829
9.73 layers/fc_layer.cpp File Reference . . . 829
9.73.1 Detailed Description . . . 830
9.74 layers/fc_layer.h File Reference . . . 830
9.74.1 Detailed Description . . . 831
9.75 layers/flatten_layer.cpp File Reference . . . 831
9.75.1 Detailed Description . . . 832
9.76 layers/flatten_layer.h File Reference . . . 832
9.76.1 Detailed Description . . . 833
9.77 layers/gru.cpp File Reference . . . 833
9.77.1 Detailed Description . . . 834
9.78 layers/gru.h File Reference . . . 834
9.78.1 Detailed Description . . . 835
9.79 layers/grucell.cpp File Reference . . . 835
9.79.1 Detailed Description . . . 836
9.80 layers/grucell.h File Reference . . . 836
9.80.1 Detailed Description . . . 837
9.81 layers/input_layer.cpp File Reference . . . 837
9.81.1 Detailed Description . . . 838
9.82 layers/input_layer.h File Reference . . . 838
9.82.1 Detailed Description . . . 839
9.83 layers/layer_context.cpp File Reference . . . 839
9.83.1 Detailed Description . . . 840
9.84 layers/layer_context.h File Reference . . . 840
9.84.1 Detailed Description . . . 841

9.85 layers/layer_devel.h File Reference . . . 841
9.85.1 Detailed Description . . . 842
9.86 layers/layer_impl.h File Reference . . . 842
9.86.1 Detailed Description . . . 843
9.87 layers/layer_node.cpp File Reference . . . 844
9.87.1 Detailed Description . . . 845
9.88 layers/layer_node.h File Reference . . . 846
9.88.1 Detailed Description . . . 847
9.89 layers/layer_normalization_layer.cpp File Reference . . . 848
9.89.1 Detailed Description . . . 848
9.90 layers/layer_normalization_layer.h File Reference . . . 849
9.90.1 Detailed Description . . . 849
9.91 layers/loss/constant_derivative_loss_layer.cpp File Reference . . . 849
9.91.1 Detailed Description . . . 850
9.92 layers/loss/cross_entropy_loss_layer.h File Reference . . . 850
9.92.1 Detailed Description . . . 851
9.93 layers/loss/cross_entropy_sigmoid_loss_layer.cpp File Reference . . . 851
9.93.1 Detailed Description . . . 852
9.94 layers/loss/cross_entropy_sigmoid_loss_layer.h File Reference . . . 852
9.94.1 Detailed Description . . . 852
9.95 layers/loss/kld_loss_layer.cpp File Reference . . . 853
9.95.1 Detailed Description . . . 853
9.96 layers/loss/kld_loss_layer.h File Reference . . . 854
9.96.1 Detailed Description . . . 854
9.97 layers/loss/loss_layer.cpp File Reference . . . 855
9.97.1 Detailed Description . . . 855
9.98 layers/loss/loss_layer.h File Reference . . . 856
9.98.1 Detailed Description . . . 856
9.99 layers/loss/mse_loss_layer.cpp File Reference . . . 857
9.99.1 Detailed Description . . . 857
9.100 layers/loss/mse_loss_layer.h File Reference . . . 858
9.100.1 Detailed Description . . . 858
9.101 layers/lstm.cpp File Reference . . . 858
9.101.1 Detailed Description . . . 859
9.102 layers/lstm.h File Reference . . . 859
9.102.1 Detailed Description . . . 860
9.103 layers/lstmcell.cpp File Reference . . . 860
9.103.1 Detailed Description . . . 861
9.104 layers/lstmcell.h File Reference . . . 861
9.104.1 Detailed Description . . . 861
9.105 layers/lstmcell_core.cpp File Reference . . . 862
9.105.1 Detailed Description . . . 862

9.106 layers/lstmcell_core.h File Reference . . . 863
9.106.1 Detailed Description . . . 863
9.107 layers/mol_attention_layer.cpp File Reference . . . 864
9.107.1 Detailed Description . . . 864
9.108 layers/mol_attention_layer.h File Reference . . . 865
9.108.1 Detailed Description . . . 865
9.109 layers/multi_head_attention_layer.cpp File Reference . . . 866
9.109.1 Detailed Description . . . 866
9.110 layers/multi_head_attention_layer.h File Reference . . . 867
9.110.1 Detailed Description . . . 867
9.111 layers/multiout_layer.cpp File Reference . . . 868
9.111.1 Detailed Description . . . 868
9.112 layers/multiout_layer.h File Reference . . . 869
9.112.1 Detailed Description . . . 869
9.113 layers/nnstreamer_layer.cpp File Reference . . . 869
9.113.1 Detailed Description . . . 870
9.114 layers/nnstreamer_layer.h File Reference . . . 871
9.114.1 Detailed Description . . . 871
9.115 layers/permute_layer.cpp File Reference . . . 872
9.115.1 Detailed Description . . . 872
9.116 layers/permute_layer.h File Reference . . . 873
9.116.1 Detailed Description . . . 874
9.117 layers/plugged_layer.h File Reference . . . 874
9.117.1 Detailed Description . . . 875
9.118 layers/pooling2d_layer.cpp File Reference . . . 875
9.118.1 Detailed Description . . . 876
9.119 layers/pooling2d_layer.h File Reference . . . 876
9.119.1 Detailed Description . . . 877
9.120 layers/positional_encoding_layer.cpp File Reference . . . 877
9.120.1 Detailed Description . . . 878
9.121 layers/positional_encoding_layer.h File Reference . . . 878
9.121.1 Detailed Description . . . 879
9.122 layers/preprocess_flip_layer.cpp File Reference . . . 879
9.122.1 Detailed Description . . . 880
9.123 layers/preprocess_flip_layer.h File Reference . . . 880
9.123.1 Detailed Description . . . 880
9.124 layers/preprocess_l2norm_layer.cpp File Reference . . . 881
9.124.1 Detailed Description . . . 881
9.125 layers/preprocess_l2norm_layer.h File Reference . . . 882
9.125.1 Detailed Description . . . 883
9.126 layers/reduce_mean_layer.cpp File Reference . . . 883
9.126.1 Detailed Description . . . 884

9.127 layers/reduce_mean_layer.h File Reference . . . 884
9.127.1 Detailed Description . . . 884
9.128 layers/reshape_layer.h File Reference . . . 885
9.128.1 Detailed Description . . . 885
9.129 layers/rnn.cpp File Reference . . . 886
9.129.1 Detailed Description . . . 886
9.130 layers/rnn.h File Reference . . . 887
9.130.1 Detailed Description . . . 887
9.131 layers/rnncell.cpp File Reference . . . 887
9.131.1 Detailed Description . . . 888
9.132 layers/rnncell.h File Reference . . . 889
9.132.1 Detailed Description . . . 889
9.133 layers/split_layer.cpp File Reference . . . 889
9.133.1 Detailed Description . . . 890
9.134 layers/split_layer.h File Reference . . . 890
9.134.1 Detailed Description . . . 891
9.135 layers/tflite_layer.cpp File Reference . . . 891
9.135.1 Detailed Description . . . 892
9.136 layers/tflite_layer.h File Reference . . . 892
9.136.1 Detailed Description . . . 893
9.137 layers/time_dist.cpp File Reference . . . 893
9.137.1 Detailed Description . . . 894
9.138 layers/time_dist.h File Reference . . . 894
9.138.1 Detailed Description . . . 894
9.139 layers/zoneout_lstmcell.cpp File Reference . . . 895
9.139.1 Detailed Description . . . 895
9.140 layers/zoneout_lstmcell.h File Reference . . . 896
9.140.1 Detailed Description . . . 896
9.141 models/dynamic_training_optimization.cpp File Reference . . . 896
9.141.1 Detailed Description . . . 897
9.142 models/dynamic_training_optimization.h File Reference . . . 898
9.142.1 Detailed Description . . . 898
9.143 models/execution_mode.h File Reference . . . 899
9.143.1 Detailed Description . . . 899
9.144 models/model_common_properties.cpp File Reference . . . 899
9.144.1 Detailed Description . . . 900
9.145 models/model_common_properties.h File Reference . . . 900
9.145.1 Detailed Description . . . 901
9.146 models/model_loader.cpp File Reference . . . 901
9.146.1 Detailed Description . . . 902
9.146.2 Macro Definition Documentation . . . 902
9.146.2.1 NN_INI_RETURN_STATUS . . . 903

9.147 models/model_loader.h File Reference . . . 903
9.147.1 Detailed Description . . . 903
9.148 models/neuralnet.cpp File Reference . . . 904
9.148.1 Detailed Description . . . 905
9.149 models/neuralnet.h File Reference . . . 905
9.149.1 Detailed Description . . . 906
9.150 nntrainer_error.h File Reference . . . 906
9.150.1 Detailed Description . . . 907
9.150.2 Macro Definition Documentation . . . 908
9.150.2.1 ESTRPIPE . . . 908
9.150.2.2 NNTR_THROW_IF . . . 908
9.150.2.3 NNTR_THROW_IF_CLEANUP . . . 908
9.150.3 Enumeration Type Documentation . . . 908
9.150.3.1 ml_error_e . . . 908
9.151 nntrainer_log.h File Reference . . . 909
9.151.1 Detailed Description . . . 910
9.151.2 Macro Definition Documentation . . . 910
9.151.2.1 ml_logd . . . 910
9.151.2.2 ml_loge . . . 911
9.151.2.3 ml_logi . . . 911
9.151.2.4 ml_logw . . . 911
9.152 nntrainer_logger.cpp File Reference . . . 912
9.152.1 Detailed Description . . . 912
9.153 nntrainer_logger.h File Reference . . . 913
9.153.1 Detailed Description . . . 913
9.154 optimizers/adam.cpp File Reference . . . 914
9.154.1 Detailed Description . . . 914
9.155 optimizers/adam.h File Reference . . . 915
9.155.1 Detailed Description . . . 915
9.156 optimizers/lr_scheduler.h File Reference . . . 915
9.156.1 Detailed Description . . . 916
9.157 optimizers/lr_scheduler_constant.cpp File Reference . . . 916
9.157.1 Detailed Description . . . 917
9.158 optimizers/lr_scheduler_constant.h File Reference . . . 917
9.158.1 Detailed Description . . . 917
9.159 optimizers/lr_scheduler_exponential.cpp File Reference . . . 918
9.159.1 Detailed Description . . . 918
9.160 optimizers/lr_scheduler_exponential.h File Reference . . . 919
9.160.1 Detailed Description . . . 919
9.161 optimizers/lr_scheduler_step.cpp File Reference . . . 919
9.161.1 Detailed Description . . . 920
9.162 optimizers/lr_scheduler_step.h File Reference . . . 921

9.162.1 Detailed Description . . . 921
9.163 optimizers/optimizer_context.h File Reference . . . 922
9.163.1 Detailed Description . . . 923
9.164 optimizers/optimizer_devel.cpp File Reference . . . 923
9.164.1 Detailed Description . . . 924
9.165 optimizers/optimizer_devel.h File Reference . . . 924
9.165.1 Detailed Description . . . 925
9.166 optimizers/optimizer_wrapped.cpp File Reference . . . 925
9.166.1 Detailed Description . . . 926
9.167 optimizers/optimizer_wrapped.h File Reference . . . 926
9.167.1 Detailed Description . . . 927
9.168 optimizers/plugged_optimizer.h File Reference . . . 927
9.168.1 Detailed Description . . . 928
9.169 optimizers/sgd.cpp File Reference . . . 929
9.169.1 Detailed Description . . . 929
9.170 optimizers/sgd.h File Reference . . . 930
9.170.1 Detailed Description . . . 930
9.171 tensor/basic_planner.cpp File Reference . . . 931
9.171.1 Detailed Description . . . 931
9.172 tensor/basic_planner.h File Reference . . . 932
9.172.1 Detailed Description . . . 933
9.173 tensor/blas_interface.cpp File Reference . . . 933
9.173.1 Detailed Description . . . 934
9.173.2 Macro Definition Documentation . . . 935
9.173.2.1 sgemv_loop . . . 935
9.174 tensor/blas_interface.h File Reference . . . 935
9.174.1 Detailed Description . . . 935
9.175 tensor/cache_elem.cpp File Reference . . . 936
9.175.1 Detailed Description . . . 936
9.176 tensor/cache_elem.h File Reference . . . 937
9.176.1 Detailed Description . . . 938
9.177 tensor/cache_loader.cpp File Reference . . . 938
9.177.1 Detailed Description . . . 939
9.178 tensor/cache_loader.h File Reference . . . 939
9.178.1 Detailed Description . . . 940
9.179 tensor/cache_pool.cpp File Reference . . . 941
9.179.1 Detailed Description . . . 941
9.180 tensor/cache_pool.h File Reference . . . 942
9.180.1 Detailed Description . . . 943
9.181 tensor/lazy_tensor.cpp File Reference . . . 943
9.181.1 Detailed Description . . . 944
9.182 tensor/lazy_tensor.h File Reference . . . 944

9.182.1 Detailed Description . . . 944
9.183 tensor/manager.cpp File Reference . . . 945
9.183.1 Detailed Description . . . 946
9.184 tensor/manager.h File Reference . . . 946
9.184.1 Detailed Description . . . 947
9.185 tensor/memory_data.h File Reference . . . 948
9.185.1 Detailed Description . . . 949
9.186 tensor/memory_planner.h File Reference . . . 949
9.186.1 Detailed Description . . . 950
9.187 tensor/memory_pool.cpp File Reference . . . 950
9.187.1 Detailed Description . . . 951
9.188 tensor/memory_pool.h File Reference . . . 951
9.188.1 Detailed Description . . . 952
9.189 tensor/optimized_v1_planner.cpp File Reference . . . 953
9.189.1 Detailed Description . . . 953
9.190 tensor/optimized_v2_planner.cpp File Reference . . . 954
9.190.1 Detailed Description . . . 955
9.191 tensor/optimized_v3_planner.cpp File Reference . . . 955
9.191.1 Detailed Description . . . 956
9.192 tensor/swap_device.cpp File Reference . . . 956
9.192.1 Detailed Description . . . 957
9.193 tensor/task.h File Reference . . . 958
9.193.1 Detailed Description . . . 959
9.194 tensor/task_executor.cpp File Reference . . . 959
9.194.1 Detailed Description . . . 960
9.195 tensor/task_executor.h File Reference . . . 960
9.195.1 Detailed Description . . . 961
9.196 tensor/tensor.cpp File Reference . . . 962
9.196.1 Detailed Description . . . 963
9.196.2 Macro Definition Documentation . . . 963
9.196.2.1 CREATE_IF_EMPTY_DIMS . . . 963
9.196.2.2 transposeloop . . . 964
9.197 tensor/tensor.h File Reference . . . 964
9.197.1 Detailed Description . . . 964
9.198 tensor/tensor_dim.cpp File Reference . . . 965
9.198.1 Detailed Description . . . 965
9.199 tensor/tensor_pool.cpp File Reference . . . 966
9.199.1 Detailed Description . . . 966
9.200 tensor/tensor_pool.h File Reference . . . 967
9.200.1 Detailed Description . . . 967
9.201 tensor/tensor_wrap_specs.h File Reference . . . 968
9.201.1 Detailed Description . . . 969

9.202 tensor/var_grad.cpp File Reference . . . 969
9.202.1 Detailed Description . . . 970
9.203 tensor/var_grad.h File Reference . . . 971
9.203.1 Detailed Description . . . 971
9.204 tensor/weight.cpp File Reference . . . 972
9.204.1 Detailed Description . . . 973
9.205 tensor/weight.h File Reference . . . 973
9.205.1 Detailed Description . . . 974
9.206 utils/base_properties.cpp File Reference . . . 974
9.206.1 Detailed Description . . . 975
9.207 utils/base_properties.h File Reference . . . 975
9.207.1 Detailed Description . . . 976
9.208 utils/ini_wrapper.cpp File Reference . . . 976
9.208.1 Detailed Description . . . 977
9.209 utils/ini_wrapper.h File Reference . . . 977
9.209.1 Detailed Description . . . 978
9.210 utils/nntr_threads.cpp File Reference . . . 978
9.210.1 Detailed Description . . . 979
9.211 utils/nntr_threads.h File Reference . . . 979
9.211.1 Detailed Description . . . 980
9.212 utils/node_exporter.cpp File Reference . . . 980
9.212.1 Detailed Description . . . 981
9.213 utils/node_exporter.h File Reference . . . 981
9.213.1 Detailed Description . . . 983
9.214 utils/profiler.cpp File Reference . . . 983
9.214.1 Detailed Description . . . 984
9.215 utils/profiler.h File Reference . . . 984
9.215.1 Detailed Description . . . 986
9.216 utils/tracer.h File Reference . . . 986
9.216.1 Detailed Description . . . 987
9.217 utils/util_func.cpp File Reference . . . 987
9.217.1 Detailed Description . . . 988
9.218 utils/util_func.h File Reference . . . 989
9.218.1 Detailed Description . . . 989

Index **991**

Chapter 1

Todo List

File concat_layer.cpp

merge concat and split layer to a common implementation

File flatten_layer.cpp

Update flatten to work in-place properly.

Update flatten to work in-place properly.

Class Layer

Check the caller of the getTensor() and set restrictions on the tensors to be accessed based on which function is requesting it.

File layer_node.h

Add printPreset support

File memory_pool.h

Support an external allocator for different backends and alignment

Support releaseMemory(token) - this need not release actual memory until deallocate

Support maximum memory size for the memory pool as an argument

support late memory request without optimization

File network_graph.h

Support multi-input graph.

File nnstreamer_layer.cpp

: provide input/output dimensions to nnstreamer for certain frameworks

: support transposing the data to support NCHW nntrainer data to NHWC nnstreamer data

Member nntrainer::buildTrasposeString (const std::array< props::PermuteDims, 3 > &arr)

deprecate this

Member nntrainer::DirDataProducer::finalize (const std::vector< TensorDim > &input_dims, const std::vector< TensorDim > &label_dims, void *user_data=nullptr) override

expand this to non onehot case

Member nntrainer::DirDataProducer::isMultiThreadSafe () const override

make this true, it is needed to test multiple worker scenario

Member nntrainer::FlatBufferOpNode::setLayerNode (const LayerNode &layer)

Now support only mse, cross

Member nntrainer::GraphCompiler::compile (std::shared_ptr< const GraphRepresentation > representation)=0

consider adding delegates argument here when implementing it for real.

Member nntrainer::GraphRealizer::realize (const GraphRepresentation &reference)=0

consider void GraphRepresentation &

Member nntrainer::IniGraphInterpreter::deserialize (const std::string &in) override

: this will be changed to a general way to add a graph

if graph Model Type is of recurrent_wrapper, parse model and realize before return

Class nntrainer::IniSection

consider API style setEntry function

Member nntrainer::InitLayerContext::requestTensor (const TensorDim &dim, const std::string &name, const Tensor::Initializer init=Tensor::Initializer::NONE, bool trainable=false, TensorLifespan lifespan=TensorLifespan::ITERATION_LIFESPAN, bool private_=true)

Consider providing a guarantee that the returned indices will always start from 0 and will always be incremental.

Member nntrainer::InitLayerContext::requestTensor (const TensorSpec &spec)

Consider providing a guarantee that the returned indices will always start from 0 and will always be incremental.

Member nntrainer::InitLayerContext::requestWeight (const TensorDim &dim, const Tensor::Initializer init, const WeightRegularizer reg, const float reg_const, const float decay, const std::string &name, bool trainable=true)

Consider providing a guarantee that the returned indices will always start from 0 and will always be incremental.

Member nntrainer::InitLayerContext::requestWeight (const WeightSpec &spec)

Consider providing a guarantee that the returned indices will always start from 0 and will always be incremental.

Class nntrainer::IterationQueue

apply this to the databuffer

handle error case: 1. when ScopedView<Sample> has met throw

1. when ScopedView<Iteration> has met throw

Member nntrainer::LossRealizer::realize (const GraphRepresentation &reference) override

support more loss layers

Member nntrainer::profile::Profiler::end (const int time_item)

: consider race condition

Member nntrainer::propagateTimestep (LayerNode *node, unsigned int time_step, unsigned int max_time_step)

add an interface to check if a layer supports a property

Member nntrainer::props::Loss::isValid (const float &v) const override

detect when loss becomes Nan is useful. But it will need dedicated throw

Class nntrainer::props::NumClass

deprecate this

Member nntrainer::RandomDataOneHotProducer::finalize (const std::vector< TensorDim > &input_dims, const std::vector< TensorDim > &label_dims, void *user_data=nullptr) override

expand this to non onehot case

move this to higher order component

Member nntrainer::RandomDataOneHotProducer::isMultiThreadSafe () const override

make this true, it is needed to test multiple worker scenario

Member nntrainer::RawFileDataProducer::pixel_size

make this a configurable type

Member nntrainer::RecurrentRealizer::realize (const GraphRepresentation &reference) override

have axis in concat layer

this has to be wrapped with identity layer as #1793

Member nntrainer::RecurrentRealizer::RecurrentRealizer (const char *ini, const std::vector< std::string > &external_input_layers)

delegate to RecurrentRealizer(

Member nntrainer::RecurrentRealizer::RecurrentRealizer (const std::vector< std::string > &properties, const std::vector< Connection > &input_conns, const std::vector< Connection > &end_conns)

Deal as sequence as proper connection with identity layer

Member nntrainer::setInplaceSharedMemoryConfigByLayer (const std::shared_ptr< LayerNode > &lnode, bool &shared_var, bool &shared_grad)

for addition layer, variables are not shared but gradients are

for layers which support in-place, both variables and gradients will be shared.

add a check here is the layer being checked here can support in-place or not

Member nntrainer::TensorSpecV2::additional_exec_order

make this as an opaque information with PIMPL

Member nntrainer::TfliteInterpreter::serialize (const GraphRepresentation &representation, const std::string &out) override

check if graph is finalized & initialized and ready to serialize.

Member nntrainer::TfOpNode::finalize ()

comment out below codes. TfOpNode needs to have LayerNode pointer

Member nntrainer::TfOpNode::setLayerNode (const LayerNode &layer)

support more loss layers

support more virtual nodes

Member nntrainer::value

make this property

Member nntrainer::Var_Grad::Var_Grad (const TensorDim &dim, const Tensor::Initializer init=Tensor::Initializer::NONE, bool ng=true, bool alloc_now=false, const std::string &name="")

gradient initializer should be none, and then they should be set zero right before using by the user itself.

Member nntrainer::WeightRegularizer

Update to TensorRegularizer

File split_layer.h

Add support for uneven splits. For now, this can be acheived with combination of split and concat layers.

File tensor.h

deprecate new tensor allocation for out of place operations.

File tensor_pool.cpp

add checks for request/updates that finalize is not done

check before allocate that finalize is done

Chapter 2

Bug List

File acti_func.cpp

No known bugs except for NYI items

No known bugs except for NYI items

File activation_layer.cpp

No known bugs except for NYI items

File activation_layer.h

No known bugs except for NYI items

File activation_realizer.cpp

No known bugs except for NYI items

File activation_realizer.h

No known bugs except for NYI items

File adam.cpp

No known bugs except for NYI items

File adam.h

No known bugs except for NYI items

File addition_layer.cpp

No known bugs except for NYI items

File addition_layer.h

No known bugs except for NYI items

File app_context.cpp

No known bugs except for NYI items

File app_context.h

No known bugs except for NYI items

File attention_layer.cpp

No known bugs except for NYI items

File attention_layer.h

No known bugs except for NYI items

File base_properties.cpp

No known bugs except for NYI items

File base_properties.h

No known bugs except for NYI items

File basic_planner.cpp

No known bugs except for NYI items

File basic_planner.h

No known bugs except for NYI items

File blas_interface.cpp

No known bugs except for NYI items

File blas_interface.h

No known bugs except for NYI items

File bn_layer.cpp

No known bugs except for NYI items

File bn_layer.h

No known bugs except for NYI items

File bn_realizer.cpp

No known bugs except for NYI items

File bn_realizer.h

No known bugs except for NYI items

File cache_elem.cpp

No known bugs except for NYI items

File cache_elem.h

No known bugs except for NYI items

File cache_loader.cpp

No known bugs except for NYI items

File cache_loader.h

No known bugs except for NYI items

File cache_pool.cpp

No known bugs except for NYI items

File cache_pool.h

No known bugs except for NYI items

File centroid_knn.cpp

No known bugs except for NYI items

File centroid_knn.h

No known bugs except for NYI items

File common_properties.cpp

No known bugs except for NYI items

File common_properties.h

No known bugs except for NYI items

File compiler.h

No known bugs except for NYI items

File compiler_fwd.h

No known bugs except for NYI items

File concat_layer.cpp

No known bugs except for NYI items

File concat_layer.h

No known bugs except for NYI items

File connection.cpp

No known bugs except for NYI items

File connection.h

No known bugs except for NYI items

File constant_derivative_loss_layer.cpp

No known bugs except for NYI items

File conv1d_layer.h

No known bugs except for NYI items

No known bugs except for NYI items

File conv2d_layer.h

No known bugs except for NYI items

No known bugs except for NYI items

File cross_entropy_loss_layer.h

No known bugs except for NYI items

File cross_entropy_sigmoid_loss_layer.cpp

No known bugs except for NYI items

File cross_entropy_sigmoid_loss_layer.h

No known bugs except for NYI items

File data_iteration.cpp

No known bugs except for NYI items

File data_iteration.h

No known bugs except for NYI items

File data_producer.h

No known bugs except for NYI items

File databuffer.cpp

No known bugs except for NYI items

File databuffer.h

No known bugs except for NYI items

File databuffer_factory.cpp

No known bugs except for NYI items

File databuffer_factory.h

No known bugs except for NYI items

File delegate.h

No known bugs except for NYI items

File dir_data_producers.h

No known bugs except for NYI items

No known bugs except for NYI items

File dropout.h

No known bugs except for NYI items

File dynamic_training_optimization.cpp

No known bugs except for NYI items

File dynamic_training_optimization.h

No known bugs except for NYI items

File embedding.cpp

No known bugs except for NYI items

File embedding.h

No known bugs except for NYI items

File execution_mode.h

No known bugs except for NYI items

File fc_layer.cpp

No known bugs except for NYI items

File fc_layer.h

No known bugs except for NYI items

File flatbuffer_interpreter.cpp

No known bugs except for NYI items

File flatbuffer_interpreter.h

No known bugs except for NYI items

File flatbuffer_opnode.h

No known bugs except for NYI items

No known bugs except for NYI items

File flatten_layer.cpp

No known bugs except for NYI items

No known bugs except for NYI items

File flatten_layer.h

No known bugs except for NYI items

File flatten_realizer.cpp

No known bugs except for NYI items

File flatten_realizer.h

No known bugs except for NYI items

File func_data_producer.cpp

No known bugs except for NYI items

File func_data_producer.h

No known bugs except for NYI items

File graph_node.h

No known bugs except for NYI items

File gru.cpp

No known bugs except for NYI items

File gru.h

No known bugs except for NYI items

File grucell.cpp

No known bugs except for NYI items

File grucell.h

No known bugs except for NYI items

File ini_interpreter.cpp

No known bugs except for NYI items

File ini_interpreter.h

No known bugs except for NYI items

File ini_wrapper.cpp

No known bugs except for NYI items

File ini_wrapper.h

No known bugs except for NYI items

File input_layer.cpp

No known bugs except for NYI items

File input_layer.h

No known bugs except for NYI items

File input_realizer.cpp

No known bugs except for NYI items

File input_realizer.h

No known bugs except for NYI items

File interpreter.h

No known bugs except for NYI items

File iteration_queue.cpp

No known bugs except for NYI items

File iteration_queue.h

No known bugs except for NYI items

File kld_loss_layer.cpp

No known bugs except for NYI items

File kld_loss_layer.h

No known bugs except for NYI items

File layer_context.cpp

No known bugs except for NYI items

File layer_context.h

No known bugs except for NYI items

File layer_devel.h

No known bugs except for NYI items

File layer_impl.h

No known bugs except for NYI items

No known bugs except for NYI items

File layer_node.cpp

No known bugs except for NYI items

File layer_node.h

No known bugs except for NYI items

File layer_normalization_layer.cpp

No known bugs except for NYI items

File layer_normalization_layer.h

No known bugs except for NYI items

File lazy_tensor.cpp

No known bugs except for NYI items

File lazy_tensor.h

No known bugs except for NYI items

File loss_layer.cpp

No known bugs except for NYI items

File loss_layer.h

No known bugs except for NYI items

No known bugs except for NYI items

File loss_realizer.h

No known bugs except for NYI items

No known bugs except for NYI items

File lr_scheduler.h

No known bugs except for NYI items

File lr_scheduler_constant.cpp

No known bugs except for NYI items

File lr_scheduler_constant.h

No known bugs except for NYI items

File lr_scheduler_exponential.cpp

No known bugs except for NYI items

File lr_scheduler_exponential.h

No known bugs except for NYI items

File lr_scheduler_step.cpp

No known bugs except for NYI items

File lr_scheduler_step.h

No known bugs except for NYI items

File lstm.cpp

No known bugs except for NYI items

File lstm.h

No known bugs except for NYI items

File lstmcell.cpp

No known bugs except for NYI items

File lstmcell.h

No known bugs except for NYI items

File lstmcell_core.cpp

No known bugs except for NYI items

File lstmcell_core.h

No known bugs except for NYI items

File manager.cpp

No known bugs except for NYI items

File manager.h

No known bugs except for NYI items

File memory_data.h

No known bugs except for NYI items

File memory_planner.h

No known bugs except for NYI items

File memory_pool.cpp

No known bugs except for NYI items

File memory_pool.h

No known bugs except for NYI items

File model_common_properties.cpp

No known bugs except for NYI items

File model_common_properties.h

No known bugs except for NYI items

File model_loader.cpp

No known bugs except for NYI items

File model_loader.h

No known bugs except for NYI items

File mol_attention_layer.cpp

No known bugs except for NYI items

File mol_attention_layer.h

No known bugs except for NYI items

File mse_loss_layer.cpp

No known bugs except for NYI items

File mse_loss_layer.h

No known bugs except for NYI items

File multi_head_attention_layer.cpp

No known bugs except for NYI items

File multi_head_attention_layer.h

No known bugs except for NYI items

File multiout_layer.cpp

No known bugs except for NYI items

File multiout_layer.h

No known bugs except for NYI items

File multiout_realizer.h

No known bugs except for NYI items

No known bugs except for NYI items

File network_graph.h

No known bugs except for NYI items

No known bugs except for NYI items

No known bugs except for NYI items

No known bugs except for NYI items

File neuralnet.cpp

No known bugs except for NYI items

File neuralnet.h

No known bugs except for NYI items

File nnstreamer_layer.cpp

No known bugs except for NYI items

File nnstreamer_layer.h

No known bugs except for NYI items

File nntr_threads.cpp

No known bugs except for NYI items

File nntr_threads.h

No known bugs except for NYI items

File nntrainer_error.h

No known bugs except for NYI items

File nntrainer_log.h

No known bugs except for NYI items

File nntrainer_logger.cpp

No known bugs except for NYI items

File nntrainer_logger.h

No known bugs except for NYI items

File node_exporter.cpp

No known bugs except for NYI items

File node_exporter.h

No known bugs except for NYI items

File optimized_v1_planner.cpp

No known bugs except for NYI items

File optimized_v2_planner.cpp

No known bugs except for NYI items

File optimized_v3_planner.cpp

No known bugs except for NYI items

File optimizer_context.h

No known bugs except for NYI items

No known bugs except for NYI items

File optimizer_devel.cpp

No known bugs except for NYI items

File optimizer_devel.h

No known bugs except for NYI items

File optimizer_wrapped.cpp

No known bugs except for NYI items

File optimizer_wrapped.h

No known bugs except for NYI items

File permute_layer.cpp

No known bugs except for NYI items

File permute_layer.h

No known bugs except for NYI items

File plugged_layer.h

No known bugs except for NYI items

File plugged_optimizer.h

No known bugs except for NYI items

File pooling2d_layer.cpp

No known bugs except for NYI items

File pooling2d_layer.h

No known bugs except for NYI items

File positional_encoding_layer.cpp

No known bugs except for NYI items

File positional_encoding_layer.h

No known bugs except for NYI items

File preprocess_flip_layer.cpp

No known bugs except for NYI items

File preprocess_flip_layer.h

No known bugs except for NYI items

File preprocess_l2norm_layer.cpp

No known bugs except for NYI items

File preprocess_l2norm_layer.h

No known bugs except for NYI items

File previous_input_realizer.cpp

No known bugs except for NYI items

File previous_input_realizer.h

No known bugs except for NYI items

File profiler.cpp

No known bugs except for NYI items

File profiler.h

No known bugs except for NYI items

File random_data_producers.cpp

No known bugs except for NYI items

File random_data_producers.h

No known bugs except for NYI items

File raw_file_data_producer.cpp

No known bugs except for NYI items

File raw_file_data_producer.h

No known bugs except for NYI items

File realizer.h

No known bugs except for NYI items

File recurrent_realizer.h

No known bugs except for NYI items

No known bugs except for NYI items

File reduce_mean_layer.cpp

No known bugs except for NYI items

File reduce_mean_layer.h

No known bugs except for NYI items

File remap_realizer.h

No known bugs except for NYI items

No known bugs except for NYI items

File reshape_layer.h

No known bugs except for NYI items

File rnn.cpp

No known bugs except for NYI items

File rnn.h

No known bugs except for NYI items

File rnncell.cpp

No known bugs except for NYI items

File rnncell.h

No known bugs except for NYI items

File sgd.cpp

No known bugs except for NYI items

File sgd.h

No known bugs except for NYI items

File slice_realizer.cpp

No known bugs except for NYI items

File slice_realizer.h

No known bugs except for NYI items

File split_layer.cpp

No known bugs except for NYI items

File split_layer.h

No known bugs except for NYI items

File swap_device.cpp

No known bugs except for NYI items

No known bugs except for NYI items

File task.h

No known bugs except for NYI items

File task_executor.cpp

No known bugs except for NYI items

File task_executor.h

No known bugs except for NYI items

File tensor.cpp

No known bugs except for NYI items

File tensor.h

No known bugs except for NYI items

File tensor_dim.cpp

No known bugs except for NYI items

File tensor_pool.cpp

No known bugs except for NYI items

File tensor_pool.h

No known bugs except for NYI items

File tensor_wrap_specs.h

No known bugs except for NYI items

File tflite_interpreter.cpp

No known bugs except for NYI items

File tflite_interpreter.h

No known bugs except for NYI items

File tflite_layer.cpp

No known bugs except for NYI items

File tflite_layer.h

No known bugs except for NYI items

File tflite_opnode.cpp

No known bugs except for NYI items

File tflite_opnode.h

No known bugs except for NYI items

File time_dist.cpp

No known bugs except for NYI items

File time_dist.h

No known bugs except for NYI items

File tracer.h

No known bugs except for NYI items

File util_func.cpp

No known bugs except for NYI items

File util_func.h

No known bugs except for NYI items

File var_grad.cpp

No known bugs except for NYI items

File var_grad.h

No known bugs except for NYI items

File weight.cpp

No known bugs except for NYI items

File weight.h

No known bugs except for NYI items

File zoneout_lstmcell.cpp

No known bugs except for NYI items

File zoneout_lstmcell.h

No known bugs except for NYI items

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

nntrainer 49
nntrainer::exception 77

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

tflite::AbsOptionsBuilder . . . 79
nntrainer::props::ActivationTypeInfo . . . 83
tflite::AddNOptionsBuilder . . . 84
tflite::AddOptionsBuilder . . . 84
nntrainer::AppContext . . . 85
tflite::ArgMaxOptionsBuilder . . . 95
tflite::ArgMinOptionsBuilder . . . 95
nntrainer::Backend . . . 101
tflite::BatchMatMulOptionsBuilder . . . 110
tflite::BatchToSpaceNDOptionsBuilder . . . 111
tflite::BidirectionalSequenceLSTMOptionsBuilder . . . 116
tflite::BidirectionalSequenceRNNOptionsBuilder . . . 116
nntrainer::Tensor::BroadcastInfo . . . 125
tflite::BufferBuilder . . . 126
tflite::BuiltinOptionsTraits< T > . . . 127
tflite::BuiltinOptionsTraits< tflite::AbsOptions > . . . 127
tflite::BuiltinOptionsTraits< tflite::AddNOptions > . . . 127
tflite::BuiltinOptionsTraits< tflite::AddOptions > . . . 128
tflite::BuiltinOptionsTraits< tflite::ArgMaxOptions > . . . 128
tflite::BuiltinOptionsTraits< tflite::ArgMinOptions > . . . 128
tflite::BuiltinOptionsTraits< tflite::BatchMatMulOptions > . . . 129
tflite::BuiltinOptionsTraits< tflite::BatchToSpaceNDOptions > . . . 129
tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceLSTMOptions > . . . 129
tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceRNNOptions > . . . 130
tflite::BuiltinOptionsTraits< tflite::CallOptions > . . . 130
tflite::BuiltinOptionsTraits< tflite::CastOptions > . . . 130
tflite::BuiltinOptionsTraits< tflite::ConcatEmbeddingsOptions > . . . 131
tflite::BuiltinOptionsTraits< tflite::ConcatenationOptions > . . . 131
tflite::BuiltinOptionsTraits< tflite::Conv2DOptions > . . . 131
tflite::BuiltinOptionsTraits< tflite::CosOptions > . . . 132
tflite::BuiltinOptionsTraits< tflite::CumsumOptions > . . . 132
tflite::BuiltinOptionsTraits< tflite::DensifyOptions > . . . 132
tflite::BuiltinOptionsTraits< tflite::DepthToSpaceOptions > . . . 133
tflite::BuiltinOptionsTraits< tflite::DepthwiseConv2DOptions > . . . 133
tflite::BuiltinOptionsTraits< tflite::DequantizeOptions > . . . 133

tflite::BuiltinOptionsTraits< tflite::DivOptions > . . . 134
tflite::BuiltinOptionsTraits< tflite::EmbeddingLookupSparseOptions > . . . 134
tflite::BuiltinOptionsTraits< tflite::EqualOptions > . . . 134
tflite::BuiltinOptionsTraits< tflite::ExpandDimsOptions > . . . 135
tflite::BuiltinOptionsTraits< tflite::ExpOptions > . . . 135
tflite::BuiltinOptionsTraits< tflite::FakeQuantOptions > . . . 135
tflite::BuiltinOptionsTraits< tflite::FillOptions > . . . 136
tflite::BuiltinOptionsTraits< tflite::FloorDivOptions > . . . 136
tflite::BuiltinOptionsTraits< tflite::FloorModOptions > . . . 136
tflite::BuiltinOptionsTraits< tflite::FullyConnectedOptions > . . . 137
tflite::BuiltinOptionsTraits< tflite::GatherNdOptions > . . . 137
tflite::BuiltinOptionsTraits< tflite::GatherOptions > . . . 137
tflite::BuiltinOptionsTraits< tflite::GreaterEqualOptions > . . . 138
tflite::BuiltinOptionsTraits< tflite::GreaterOptions > . . . 138
tflite::BuiltinOptionsTraits< tflite::HardSwishOptions > . . . 138
tflite::BuiltinOptionsTraits< tflite::IfOptions > . . . 139
tflite::BuiltinOptionsTraits< tflite::L2NormOptions > . . . 139
tflite::BuiltinOptionsTraits< tflite::LeakyReluOptions > . . . 139
tflite::BuiltinOptionsTraits< tflite::LessEqualOptions > . . . 140
tflite::BuiltinOptionsTraits< tflite::LessOptions > . . . 140
tflite::BuiltinOptionsTraits< tflite::LocalResponseNormalizationOptions > . . . 140
tflite::BuiltinOptionsTraits< tflite::LogicalAndOptions > . . . 141
tflite::BuiltinOptionsTraits< tflite::LogicalNotOptions > . . . 141
tflite::BuiltinOptionsTraits< tflite::LogicalOrOptions > . . . 141
tflite::BuiltinOptionsTraits< tflite::LogSoftmaxOptions > . . . 142
tflite::BuiltinOptionsTraits< tflite::LSHProjectionOptions > . . . 142
tflite::BuiltinOptionsTraits< tflite::LSTMOptions > . . . 142
tflite::BuiltinOptionsTraits< tflite::MatrixDiagOptions > . . . 143
tflite::BuiltinOptionsTraits< tflite::MatrixSetDiagOptions > . . . 143
tflite::BuiltinOptionsTraits< tflite::MaximumMinimumOptions > . . . 143
tflite::BuiltinOptionsTraits< tflite::MirrorPadOptions > . . . 144
tflite::BuiltinOptionsTraits< tflite::MulOptions > . . . 144
tflite::BuiltinOptionsTraits< tflite::NegOptions > . . . 144
tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV4Options > . . . 145
tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV5Options > . . . 145
tflite::BuiltinOptionsTraits< tflite::NotEqualOptions > . . . 145
tflite::BuiltinOptionsTraits< tflite::OneHotOptions > . . . 146
tflite::BuiltinOptionsTraits< tflite::PackOptions > . . . 146
tflite::BuiltinOptionsTraits< tflite::PadOptions > . . . 146
tflite::BuiltinOptionsTraits< tflite::PadV2Options > . . . 147
tflite::BuiltinOptionsTraits< tflite::Pool2DOptions > . . . 147
tflite::BuiltinOptionsTraits< tflite::PowOptions > . . . 147
tflite::BuiltinOptionsTraits< tflite::QuantizeOptions > . . . 148
tflite::BuiltinOptionsTraits< tflite::RangeOptions > . . . 148
tflite::BuiltinOptionsTraits< tflite::RankOptions > . . . 148
tflite::BuiltinOptionsTraits< tflite::ReducerOptions > . . . 149
tflite::BuiltinOptionsTraits< tflite::ReshapeOptions > . . . 149
tflite::BuiltinOptionsTraits< tflite::ResizeBilinearOptions > . . . 149
tflite::BuiltinOptionsTraits< tflite::ResizeNearestNeighborOptions > . . . 150
tflite::BuiltinOptionsTraits< tflite::ReverseSequenceOptions > . . . 150
tflite::BuiltinOptionsTraits< tflite::ReverseV2Options > . . . 150
tflite::BuiltinOptionsTraits< tflite::RNNOptions > . . . 151
tflite::BuiltinOptionsTraits< tflite::ScatterNdOptions > . . . 151
tflite::BuiltinOptionsTraits< tflite::SegmentSumOptions > . . . 151
tflite::BuiltinOptionsTraits< tflite::SelectOptions > . . . 152
tflite::BuiltinOptionsTraits< tflite::SelectV2Options > . . . 152
tflite::BuiltinOptionsTraits< tflite::SequenceRNNOptions > . . . 152
tflite::BuiltinOptionsTraits< tflite::ShapeOptions > . . . 153

tflite::BuiltinOptionsTraits< tflite::SkipGramOptions > . . . 153
tflite::BuiltinOptionsTraits< tflite::SliceOptions > . . . 153
tflite::BuiltinOptionsTraits< tflite::SoftmaxOptions > . . . 154
tflite::BuiltinOptionsTraits< tflite::SpaceToBatchNDOptions > . . . 154
tflite::BuiltinOptionsTraits< tflite::SpaceToDepthOptions > . . . 154
tflite::BuiltinOptionsTraits< tflite::SparseToDenseOptions > . . . 155
tflite::BuiltinOptionsTraits< tflite::SplitOptions > . . . 155
tflite::BuiltinOptionsTraits< tflite::SplitVOptions > . . . 155
tflite::BuiltinOptionsTraits< tflite::SquaredDifferenceOptions > . . . 156
tflite::BuiltinOptionsTraits< tflite::SquareOptions > . . . 156
tflite::BuiltinOptionsTraits< tflite::SqueezeOptions > . . . 156
tflite::BuiltinOptionsTraits< tflite::StridedSliceOptions > . . . 157
tflite::BuiltinOptionsTraits< tflite::SubOptions > . . . 157
tflite::BuiltinOptionsTraits< tflite::SVDFOptions > . . . 157
tflite::BuiltinOptionsTraits< tflite::TileOptions > . . . 158
tflite::BuiltinOptionsTraits< tflite::TopKV2Options > . . . 158
tflite::BuiltinOptionsTraits< tflite::TransposeConvOptions > . . . 158
tflite::BuiltinOptionsTraits< tflite::TransposeOptions > . . . 159
tflite::BuiltinOptionsTraits< tflite::UnidirectionalSequenceLSTMOptions > . . . 159
tflite::BuiltinOptionsTraits< tflite::UniqueOptions > . . . 159
tflite::BuiltinOptionsTraits< tflite::UnpackOptions > . . . 160
tflite::BuiltinOptionsTraits< tflite::WhereOptions > . . . 160
tflite::BuiltinOptionsTraits< tflite::WhileOptions > . . . 160
tflite::BuiltinOptionsTraits< tflite::ZerosLikeOptions > . . . 161
nntrainer::CacheElem . . . 161
nntrainer::CacheLoader . . . 164
tflite::CallOptionsBuilder . . . 180
tflite::CastOptionsBuilder . . . 181
tflite::ConcatEmbeddingsOptionsBuilder . . . 190
tflite::ConcatenationOptionsBuilder . . . 190
nntrainer::Connection . . . 191
nntrainer::props::connection_prop_tag . . . 197
tflite::Conv2DOptionsBuilder . . . 197
tflite::CosOptionsBuilder . . . 198
tflite::CumsumOptionsBuilder . . . 198
tflite::CustomQuantizationBuilder . . . 199
nntrainer::DataProducer . . . 199
  nntrainer::DirDataProducer . . . 217
  nntrainer::FuncDataProducer . . . 278
  nntrainer::RandomDataOneHotProducer . . . 547
  nntrainer::RawFileDataProducer . . . 554
nntrainer::DelegateConfig . . . 208
tflite::DensifyOptionsBuilder . . . 208
tflite::DepthToSpaceOptionsBuilder . . . 209
tflite::DepthwiseConv2DOptionsBuilder . . . 210
tflite::DequantizeOptionsBuilder . . . 210
nntrainer::Device . . . 211
tflite::DimensionMetadataBuilder . . . 216
tflite::DivOptionsBuilder . . . 227
tflite::EmbeddingLookupSparseOptionsBuilder . . . 232
EnumProperty
  nntrainer::props::Activation . . . 80
  nntrainer::props::BasicRegularizer . . . 106
    nntrainer::props::WeightRegularizer . . . 728
  nntrainer::props::BiasInitializer . . . 113
  nntrainer::props::BNPARAMS_BETA_INIT . . . 117
  nntrainer::props::BNPARAMS_GAMMA_INIT . . . 118
  nntrainer::props::BNPARAMS_MU_INIT . . . 120

nntrainer::props::BNPARAMS_VAR_INIT . . . 121
nntrainer::props::FlipDirection . . . 274
nntrainer::props::HiddenStateActivation . . . 311
nntrainer::props::PoolingType . . . 506
nntrainer::props::RecurrentActivation . . . 559
nntrainer::props::ReturnAttentionWeight . . . 577
nntrainer::props::WeightInitializer . . . 727
tflite::EqualOptionsBuilder . . . 235
nntrainer::exception::ErrorNotification< Err, > . . . 235
tflite::ExpandDimsOptionsBuilder . . . 237
tflite::ExpOptionsBuilder . . . 238
nntrainer::Exporter . . . 238
External . . . 241
tflite::FakeQuantOptionsBuilder . . . 242
tflite::FillOptionsBuilder . . . 246
nntrainer::FlatBufferOpNode . . . 251
nntrainer::props::FlipDirectionInfo . . . 275
tflite::FloorDivOptionsBuilder . . . 276
tflite::FloorModOptionsBuilder . . . 276
tflite::FullyConnectedOptionsBuilder . . . 277
tflite::GatherNdOptionsBuilder . . . 281
tflite::GatherOptionsBuilder . . . 282
nntrainer::GraphCompiler . . . 288
nntrainer::GraphInterpreter . . . 289
nntrainer::FlatBufferInterpreter . . . 249
nntrainer::IniGraphInterpreter . . . 315
nntrainer::TfliteInterpreter . . . 670
nntrainer::GraphNode . . . 292
nntrainer::LayerNode . . . 377
nntrainer::GraphRealizer . . . 306
nntrainer::ActivationRealizer . . . 81
nntrainer::BnRealizer . . . 122
nntrainer::FlattenRealizer . . . 272
nntrainer::InputRealizer . . . 354
nntrainer::LossRealizer . . . 420
nntrainer::MultioutRealizer . . . 447
nntrainer::PreviousInputRealizer . . . 516
nntrainer::RecurrentRealizer . . . 564
nntrainer::RemapRealizer . . . 571
nntrainer::SliceRealizer . . . 626
tflite::GreaterEqualOptionsBuilder . . . 308
tflite::GreaterOptionsBuilder . . . 309
tflite::HardSwishOptionsBuilder . . . 309
std::hash< nntrainer::Connection > . . . 310
tflite::IfOptionsBuilder . . . 313
nntrainer::IniSection . . . 318
nntrainer::props::InitializerInfo . . . 329
nntrainer::InitLayerContext . . . 330
nntrainer::IniWrapper . . . 341
tflite::Int32VectorBuilder . . . 358
invalid_argument
nntrainer::exception::not_supported . . . 456
nntrainer::exception::permission_denied . . . 488
nntrainer::Iteration . . . 361
nntrainer::IterationQueue . . . 368
iterator
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType > . . . 296

tflite::L2NormOptionsBuilder . . . 373
Layer . . . 377
nntrainer::CentroidKNN . . . 182
nntrainer::PermuteLayer . . . 492
nntrainer::PreprocessL2NormLayer . . . 512
Layer . . . 377
Layer . . . 377
Layer
nntrainer::internal::PluggedLayer . . . 496
Layer
nntrainer::LayerNode . . . 377
tflite::LeakyReluOptionsBuilder . . . 411
tflite::LessEqualOptionsBuilder . . . 413
tflite::LessOptionsBuilder . . . 413
tflite::LocalResponseNormalizationOptionsBuilder . . . 414
tflite::LogicalAndOptionsBuilder . . . 415
tflite::LogicalNotOptionsBuilder . . . 415
tflite::LogicalOrOptionsBuilder . . . 416
tflite::LogSoftmaxOptionsBuilder . . . 416
tflite::LSHProjectionOptionsBuilder . . . 423
tflite::LSTMOptionsBuilder . . . 423
tflite::MatrixDiagOptionsBuilder . . . 424
tflite::MatrixSetDiagOptionsBuilder . . . 424
tflite::MaximumMinimumOptionsBuilder . . . 425
nntrainer::MemoryData< T > . . . 427
nntrainer::MemoryPlanner . . . 429
nntrainer::BasicPlanner . . . 103
nntrainer::OptimizedV1Planner . . . 464
nntrainer::OptimizedV2Planner . . . 466
nntrainer::OptimizedV3Planner . . . 469
nntrainer::MemoryPool . . . 431
nntrainer::CachePool . . . 168
nntrainer::MemoryRequest . . . 438
tflite::MetadataBuilder . . . 440
tflite::MirrorPadOptionsBuilder . . . 441
tflite::ModelBuilder . . . 441
tflite::MulOptionsBuilder . . . 446
tflite::NegOptionsBuilder . . . 452
tflite::NonMaxSuppressionV4OptionsBuilder . . . 453
tflite::NonMaxSuppressionV5OptionsBuilder . . . 453
tflite::NotEqualOptionsBuilder . . . 457
tflite::OneHotOptionsBuilder . . . 461
Op . . . 462
tflite::OperatorBuilder . . . 462
tflite::OperatorCodeBuilder . . . 463
Optimizer
nntrainer::internal::PluggedOptimizer . . . 501
tflite::PackOptionsBuilder . . . 477
tflite::PadOptionsBuilder . . . 485
tflite::PadV2OptionsBuilder . . . 485
nntrainer::ParallelBatch . . . 486
tflite::Pool2DOptionsBuilder . . . 506
nntrainer::props::PoolingTypeInfo . . . 507
PositiveIntegerProperty
nntrainer::props::Axis . . . 99
nntrainer::props::SplitDimension . . . 633
nntrainer::props::ConcatDimension . . . 188

nntrainer::props::ReduceDimension . . . 568
nntrainer::props::Dilation . . . 214
nntrainer::props::FilterSize . . . 247
nntrainer::props::InDim . . . 313
nntrainer::props::KernelSize . . . 372
nntrainer::props::MaxTimestep . . . 426
nntrainer::props::MoL_K . . . 442
nntrainer::props::OutDim . . . 471
nntrainer::props::OutputShape . . . 475
nntrainer::props::PoolSize . . . 509
nntrainer::props::ProjectedKeyDim . . . 527
nntrainer::props::ProjectedValueDim . . . 529
nntrainer::props::SplitNumber . . . 635
nntrainer::props::Stride . . . 642
nntrainer::props::Unit . . . 692
nntrainer::props::UnrollFor . . . 695
nntrainer::PropsBufferSize . . . 531
PositiveIntegerProperty
nntrainer::props::DecaySteps . . . 206
nntrainer::props::NumHeads . . . 459
tflite::PowOptionsBuilder . . . 511
nntrainer::profile::ProfileEventData . . . 518
nntrainer::profile::ProfileListener . . . 519
nntrainer::profile::GenericProfileListener . . . 282
nntrainer::profile::Profiler . . . 522
Property
nntrainer::props::AsSequence . . . 96
nntrainer::props::AverageAttentionWeight . . . 97
nntrainer::props::BasicRegularizerConstant . . . 107
nntrainer::props::BiasDecay . . . 111
nntrainer::props::WeightDecay . . . 725
nntrainer::props::WeightRegularizerConstant . . . 730
nntrainer::props::Bidirectional . . . 114
nntrainer::props::ClipGradByGlobalNorm . . . 186
nntrainer::props::DecayRate . . . 204
nntrainer::props::DirPath . . . 221
nntrainer::props::DisableBias . . . 224
nntrainer::props::Distribute . . . 226
nntrainer::props::DropOutRate . . . 228
nntrainer::props::DynamicTimeSequence . . . 230
nntrainer::props::Epsilon . . . 232
nntrainer::props::FilePath . . . 243
nntrainer::props::Flatten . . . 270
nntrainer::props::GenericShape . . . 285
nntrainer::props::InputShape . . . 356
nntrainer::props::TargetShape . . . 650
nntrainer::props::InputConnection . . . 350
nntrainer::props::IntegrateBias . . . 359
nntrainer::props::Iteration . . . 366
nntrainer::props::LearningRate . . . 411
nntrainer::props::Loss . . . 417
nntrainer::props::Momentum . . . 444
nntrainer::props::Name . . . 449
nntrainer::props::InputIsSequence . . . 353
nntrainer::props::LabelLayer . . . 374
nntrainer::props::OutputLayer . . . 473
nntrainer::props::SharedFrom . . . 622
nntrainer::props::Normalization . . . 454

nntrainer::props::NumClass . . . 457
nntrainer::props::Padding1D . . . 478
nntrainer::props::Padding2D . . . 481
nntrainer::props::Padding_ . . . 484
nntrainer::props::PermuteDims . . . 489
nntrainer::props::PropsUserData . . . 544
nntrainer::props::RandomTranslate . . . 551
nntrainer::props::RecurrentInput . . . 560
nntrainer::props::RecurrentOutput . . . 562
nntrainer::props::ResetAfter . . . 573
nntrainer::props::ReturnSequences . . . 581
nntrainer::props::Standardization . . . 640
nntrainer::props::Timestep . . . 685
nntrainer::props::Trainable . . . 687
nntrainer::props::ZeroIdxMask . . . 736
nntrainer::PropsMax . . . 533
nntrainer::PropsMin . . . 535
nntrainer::PropsNNSModelPath . . . 537
nntrainer::PropsNumSamples . . . 539
nntrainer::PropsTflModelPath . . . 541
tflite::QuantizationDetailsTraits< T > . . . 545
tflite::QuantizationDetailsTraits< tflite::CustomQuantization > . . . 545
tflite::QuantizationParametersBuilder . . . 546
tflite::QuantizeOptionsBuilder . . . 546
tflite::RangeOptionsBuilder . . . 553
tflite::RankOptionsBuilder . . . 554
tflite::ReducerOptionsBuilder . . . 569
nntrainer::props::RegularizerInfo . . . 569
tflite::ReshapeOptionsBuilder . . . 575
tflite::ResizeBilinearOptionsBuilder . . . 576
tflite::ResizeNearestNeighborOptionsBuilder . . . 577
nntrainer::props::ReturnAttentionWeightInfo . . . 579
reverse_iterator
nntrainer::GraphNodeReverseIterator< T_iterator > . . . 304
tflite::ReverseSequenceOptionsBuilder . . . 582
tflite::ReverseV2OptionsBuilder . . . 583
tflite::RNNOptionsBuilder . . . 583
nntrainer::RunLayerContext . . . 584
nntrainer::RunOptimizerContext . . . 610
nntrainer::Sample . . . 614
tflite::ScatterNdOptionsBuilder . . . 616
nntrainer::ScopedView< T > . . . 616
tflite::SegmentSumOptionsBuilder . . . 619
tflite::SelectOptionsBuilder . . . 619
tflite::SelectV2OptionsBuilder . . . 620
tflite::SequenceRNNOptionsBuilder . . . 621
tflite::ShapeOptionsBuilder . . . 621
tflite::SignatureDefBuilder . . . 624
tflite::SkipGramOptionsBuilder . . . 624
tflite::SliceOptionsBuilder . . . 625
tflite::SoftmaxOptionsBuilder . . . 628
tflite::SpaceToBatchNDOptionsBuilder . . . 629
tflite::SpaceToDepthOptionsBuilder . . . 630
tflite::SparseIndexVectorTraits< T > . . . 630
tflite::SparseIndexVectorTraits< tflite::Int32Vector > . . . 631
tflite::SparseIndexVectorTraits< tflite::Uint16Vector > . . . 631
tflite::SparseIndexVectorTraits< tflite::Uint8Vector > . . . 631
tflite::SparseToDenseOptionsBuilder . . . 632

tflite::SparsityParametersBuilder . . . 632
tflite::SplitOptionsBuilder . . . 637
tflite::SplitVOptionsBuilder . . . 637
tflite::SquaredDifferenceOptionsBuilder . . . 638
tflite::SquareOptionsBuilder . . . 638
tflite::SqueezeOptionsBuilder . . . 639
nntrainer::SrcSharedTensor . . . 639
tflite::StridedSliceOptionsBuilder . . . 644
tflite::SubGraphBuilder . . . 644
tflite::SubOptionsBuilder . . . 645
tflite::SVDFOptionsBuilder . . . 645
nntrainer::SwapDevice . . . 646
Table
  tflite::FLATBUFFERS_FINAL_CLASS . . . 259
nntrainer::Task . . . 652
  nntrainer::TaskAsync< T > . . . 656
nntrainer::TaskExecutor . . . 659
tflite::TensorBuilder . . . 665
tflite::TensorMapBuilder . . . 666
nntrainer::TensorSpecV2 . . . 666
nntrainer::TfOpNode . . . 673
tflite::TileOptionsBuilder . . . 684
tflite::TopKV2OptionsBuilder . . . 686
tflite::TransposeConvOptionsBuilder . . . 689
tflite::TransposeOptionsBuilder . . . 689
tflite::Uint16VectorBuilder . . . 690
tflite::Uint8VectorBuilder . . . 690
tflite::UnidirectionalSequenceLSTMOptionsBuilder . . . 691
tflite::UniqueOptionsBuilder . . . 692
tflite::UnpackOptionsBuilder . . . 694
nntrainer::Var_Grad . . . 696
  nntrainer::Weight . . . 715
nntrainer::VarGradSpecV2 . . . 709
nntrainer::ViewQueue< T > . . . 712
nntrainer::ViewQueue< MarkableIteration > . . . 712
nntrainer::WeightSpecV2 . . . 732
nntrainer::WGradMemoryRequest . . . 734
tflite::WhereOptionsBuilder . . . 734
tflite::WhileOptionsBuilder . . . 735
tflite::ZerosLikeOptionsBuilder . . . 737

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

tflite::AbsOptionsBuilder . . . 79
nntrainer::props::Activation
Activation Enumeration Information . . . 80
nntrainer::ActivationRealizer
Graph realizer which realizes activation . . . 81
nntrainer::props::ActivationTypeInfo
Enumeration of activation function type . . . 83
tflite::AddNOptionsBuilder . . . 84
tflite::AddOptionsBuilder . . . 84
nntrainer::AppContext
App . . . 85
tflite::ArgMaxOptionsBuilder . . . 95
tflite::ArgMinOptionsBuilder . . . 95
nntrainer::props::AsSequence
Identifiers to locate a connection which should be returned as whole used in recurrent realizer 96
nntrainer::props::AverageAttentionWeight
AverageAttentionWeight, average attention weight . . . 97
nntrainer::props::Axis
Axis property, idx in the dimension . . . 99
nntrainer::Backend
Backend to be used for the operations to use . . . 101
nntrainer::BasicPlanner
Basic Memory Planner provides the basic plan for memory layout . . . 103
nntrainer::props::BasicRegularizer
BasicRegularizer Regularization Enumeration Information . . . 106
nntrainer::props::BasicRegularizerConstant
BasicRegularizerConstant property, this defines how much regularize the weight . . . 107
tflite::BatchMatMulOptionsBuilder . . . 110
tflite::BatchToSpaceNDOptionsBuilder . . . 111
nntrainer::props::BiasDecay
BiasDecay property, this defines how much regularize the weight . . . 111
nntrainer::props::BiasInitializer
BiasInitializer Initialization Enumeration Information . . . 113
nntrainer::props::Bidirectional
Bidirectional property, used to make bidirectional layers . . . 114

tflite::BidirectionalSequenceLSTMOptionsBuilder . . . 116
tflite::BidirectionalSequenceRNNOptionsBuilder . . . 116
nntrainer::props::BNPARAMS_BETA_INIT
BNPARAMS_BETA_INIT Initialization Enumeration Information . . . 117
nntrainer::props::BNPARAMS_GAMMA_INIT
BNPARAMS_GAMMA_INIT Initialization Enumeration Information . . . 118
nntrainer::props::BNPARAMS_MU_INIT
BNPARAMS_MU_INIT Initialization Enumeration Information . . . 120
nntrainer::props::BNPARAMS_VAR_INIT
BNPARAMS_VAR_INIT Initialization Enumeration Information . . . 121
nntrainer::BnRealizer
Graph realizer class which removes batch normalization layer from the graph . . . 122
nntrainer::Tensor::BroadcastInfo . . . 125
tflite::BufferBuilder . . . 126
tflite::BuiltinOptionsTraits< T > . . . 127
tflite::BuiltinOptionsTraits< tflite::AbsOptions > . . . 127
tflite::BuiltinOptionsTraits< tflite::AddNOptions > . . . 127
tflite::BuiltinOptionsTraits< tflite::AddOptions > . . . 128
tflite::BuiltinOptionsTraits< tflite::ArgMaxOptions > . . . 128
tflite::BuiltinOptionsTraits< tflite::ArgMinOptions > . . . 128
tflite::BuiltinOptionsTraits< tflite::BatchMatMulOptions > . . . 129
tflite::BuiltinOptionsTraits< tflite::BatchToSpaceNDOptions > . . . 129
tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceLSTMOptions > . . . 129
tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceRNNOptions > . . . 130
tflite::BuiltinOptionsTraits< tflite::CallOptions > . . . 130
tflite::BuiltinOptionsTraits< tflite::CastOptions > . . . 130
tflite::BuiltinOptionsTraits< tflite::ConcatEmbeddingsOptions > . . . 131
tflite::BuiltinOptionsTraits< tflite::ConcatenationOptions > . . . 131
tflite::BuiltinOptionsTraits< tflite::Conv2DOptions > . . . 131
tflite::BuiltinOptionsTraits< tflite::CosOptions > . . . 132
tflite::BuiltinOptionsTraits< tflite::CumsumOptions > . . . 132
tflite::BuiltinOptionsTraits< tflite::DensifyOptions > . . . 132
tflite::BuiltinOptionsTraits< tflite::DepthToSpaceOptions > . . . 133
tflite::BuiltinOptionsTraits< tflite::DepthwiseConv2DOptions > . . . 133
tflite::BuiltinOptionsTraits< tflite::DequantizeOptions > . . . 133
tflite::BuiltinOptionsTraits< tflite::DivOptions > . . . 134
tflite::BuiltinOptionsTraits< tflite::EmbeddingLookupSparseOptions > . . . 134
tflite::BuiltinOptionsTraits< tflite::EqualOptions > . . . 134
tflite::BuiltinOptionsTraits< tflite::ExpandDimsOptions > . . . 135
tflite::BuiltinOptionsTraits< tflite::ExpOptions > . . . 135
tflite::BuiltinOptionsTraits< tflite::FakeQuantOptions > . . . 135
tflite::BuiltinOptionsTraits< tflite::FillOptions > . . . 136
tflite::BuiltinOptionsTraits< tflite::FloorDivOptions > . . . 136
tflite::BuiltinOptionsTraits< tflite::FloorModOptions > . . . 136
tflite::BuiltinOptionsTraits< tflite::FullyConnectedOptions > . . . 137
tflite::BuiltinOptionsTraits< tflite::GatherNdOptions > . . . 137
tflite::BuiltinOptionsTraits< tflite::GatherOptions > . . . 137
tflite::BuiltinOptionsTraits< tflite::GreaterEqualOptions > . . . 138
tflite::BuiltinOptionsTraits< tflite::GreaterOptions > . . . 138
tflite::BuiltinOptionsTraits< tflite::HardSwishOptions > . . . 138
tflite::BuiltinOptionsTraits< tflite::IfOptions > . . . 139
tflite::BuiltinOptionsTraits< tflite::L2NormOptions > . . . 139
tflite::BuiltinOptionsTraits< tflite::LeakyReluOptions > . . . 139
tflite::BuiltinOptionsTraits< tflite::LessEqualOptions > . . . 140
tflite::BuiltinOptionsTraits< tflite::LessOptions > . . . 140
tflite::BuiltinOptionsTraits< tflite::LocalResponseNormalizationOptions > . . . 140
tflite::BuiltinOptionsTraits< tflite::LogicalAndOptions > . . . 141
tflite::BuiltinOptionsTraits< tflite::LogicalNotOptions > . . . 141

tflite::BuiltinOptionsTraits< tflite::LogicalOrOptions > . . . 141
tflite::BuiltinOptionsTraits< tflite::LogSoftmaxOptions > . . . 142
tflite::BuiltinOptionsTraits< tflite::LSHProjectionOptions > . . . 142
tflite::BuiltinOptionsTraits< tflite::LSTMOptions > . . . 142
tflite::BuiltinOptionsTraits< tflite::MatrixDiagOptions > . . . 143
tflite::BuiltinOptionsTraits< tflite::MatrixSetDiagOptions > . . . 143
tflite::BuiltinOptionsTraits< tflite::MaximumMinimumOptions > . . . 143
tflite::BuiltinOptionsTraits< tflite::MirrorPadOptions > . . . 144
tflite::BuiltinOptionsTraits< tflite::MulOptions > . . . 144
tflite::BuiltinOptionsTraits< tflite::NegOptions > . . . 144
tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV4Options > . . . 145
tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV5Options > . . . 145
tflite::BuiltinOptionsTraits< tflite::NotEqualOptions > . . . 145
tflite::BuiltinOptionsTraits< tflite::OneHotOptions > . . . 146
tflite::BuiltinOptionsTraits< tflite::PackOptions > . . . 146
tflite::BuiltinOptionsTraits< tflite::PadOptions > . . . 146
tflite::BuiltinOptionsTraits< tflite::PadV2Options > . . . 147
tflite::BuiltinOptionsTraits< tflite::Pool2DOptions > . . . 147
tflite::BuiltinOptionsTraits< tflite::PowOptions > . . . 147
tflite::BuiltinOptionsTraits< tflite::QuantizeOptions > . . . 148
tflite::BuiltinOptionsTraits< tflite::RangeOptions > . . . 148
tflite::BuiltinOptionsTraits< tflite::RankOptions > . . . 148
tflite::BuiltinOptionsTraits< tflite::ReducerOptions > . . . 149
tflite::BuiltinOptionsTraits< tflite::ReshapeOptions > . . . 149
tflite::BuiltinOptionsTraits< tflite::ResizeBilinearOptions > . . . 149
tflite::BuiltinOptionsTraits< tflite::ResizeNearestNeighborOptions > . . . 150
tflite::BuiltinOptionsTraits< tflite::ReverseSequenceOptions > . . . 150
tflite::BuiltinOptionsTraits< tflite::ReverseV2Options > . . . 150
tflite::BuiltinOptionsTraits< tflite::RNNOptions > . . . 151
tflite::BuiltinOptionsTraits< tflite::ScatterNdOptions > . . . 151
tflite::BuiltinOptionsTraits< tflite::SegmentSumOptions > . . . 151
tflite::BuiltinOptionsTraits< tflite::SelectOptions > . . . 152
tflite::BuiltinOptionsTraits< tflite::SelectV2Options > . . . 152
tflite::BuiltinOptionsTraits< tflite::SequenceRNNOptions > . . . 152
tflite::BuiltinOptionsTraits< tflite::ShapeOptions > . . . 153
tflite::BuiltinOptionsTraits< tflite::SkipGramOptions > . . . 153
tflite::BuiltinOptionsTraits< tflite::SliceOptions > . . . 153
tflite::BuiltinOptionsTraits< tflite::SoftmaxOptions > . . . 154
tflite::BuiltinOptionsTraits< tflite::SpaceToBatchNDOptions > . . . 154
tflite::BuiltinOptionsTraits< tflite::SpaceToDepthOptions > . . . 154
tflite::BuiltinOptionsTraits< tflite::SparseToDenseOptions > . . . 155
tflite::BuiltinOptionsTraits< tflite::SplitOptions > . . . 155
tflite::BuiltinOptionsTraits< tflite::SplitVOptions > . . . 155
tflite::BuiltinOptionsTraits< tflite::SquaredDifferenceOptions > . . . 156
tflite::BuiltinOptionsTraits< tflite::SquareOptions > . . . 156
tflite::BuiltinOptionsTraits< tflite::SqueezeOptions > . . . 156
tflite::BuiltinOptionsTraits< tflite::StridedSliceOptions > . . . 157
tflite::BuiltinOptionsTraits< tflite::SubOptions > . . . 157
tflite::BuiltinOptionsTraits< tflite::SVDFOptions > . . . 157
tflite::BuiltinOptionsTraits< tflite::TileOptions > . . . 158
tflite::BuiltinOptionsTraits< tflite::TopKV2Options > . . . 158
tflite::BuiltinOptionsTraits< tflite::TransposeConvOptions > . . . 158
tflite::BuiltinOptionsTraits< tflite::TransposeOptions > . . . 159
tflite::BuiltinOptionsTraits< tflite::UnidirectionalSequenceLSTMOptions > . . . 159
tflite::BuiltinOptionsTraits< tflite::UniqueOptions > . . . 159
tflite::BuiltinOptionsTraits< tflite::UnpackOptions > . . . 160
tflite::BuiltinOptionsTraits< tflite::WhereOptions > . . . 160
tflite::BuiltinOptionsTraits< tflite::WhileOptions > . . . 160

tflite::BuiltinOptionsTraits< tflite::ZerosLikeOptions > . . . 161
nntrainer::CacheElem
Cache element containing swap address . . . 161
nntrainer::CacheLoader
Cache loader from swap device . . . 164
nntrainer::CachePool
Cache memory with fixed size utilizing swap device . . . 168
tflite::CallOptionsBuilder . . . 180
tflite::CastOptionsBuilder . . . 181
nntrainer::CentroidKNN
Centroid KNN layer which takes centroid and do k-nearest neighbor classification . . . 182
nntrainer::props::ClipGradByGlobalNorm
Properties for getting the clipping value to clip the gradient by norm . . . 186
nntrainer::props::ConcatDimension
ConcatDimension property, dimension along which to concat the input . . . 188
tflite::ConcatEmbeddingsOptionsBuilder . . . 190
tflite::ConcatenationOptionsBuilder . . . 190
nntrainer::Connection
RAII class to define a connection . . . 191
nntrainer::props::connection_prop_tag
Connection prop tag type . . . 197
tflite::Conv2DOptionsBuilder . . . 197
tflite::CosOptionsBuilder . . . 198
tflite::CumsumOptionsBuilder . . . 198
tflite::CustomQuantizationBuilder . . . 199
nntrainer::DataProducer
DataProducer interface used to abstract data provider . . . 199
nntrainer::props::DecayRate
Decay rate property . . . 204
nntrainer::props::DecaySteps
Decay steps property . . . 206
nntrainer::DelegateConfig
Configuration for the delegate . . . 208
tflite::DensifyOptionsBuilder . . . 208
tflite::DepthToSpaceOptionsBuilder . . . 209
tflite::DepthwiseConv2DOptionsBuilder . . . 210
tflite::DequantizeOptionsBuilder . . . 210
nntrainer::Device
Device to be used for the operations to run . . . 211
nntrainer::props::Dilation
Dilation property, dilation indicates how many space will be inserted between kernel element . 214
tflite::DimensionMetadataBuilder . . . 216
nntrainer::DirDataProducer
DirDataProducer which generates a onehot vector as a label . . . 217
nntrainer::props::DirPath
Props containing directory path value . . . 221
nntrainer::props::DisableBias
DisableBias to disable the bias . . . 224
nntrainer::props::Distribute
Distribute property, true if it distribute across layer . . . 226
tflite::DivOptionsBuilder . . . 227
nntrainer::props::DropOutRate
DropOutRate property, this defines drop out specification of layer . . . 228
nntrainer::props::DynamicTimeSequence
Dynamic time sequence property, use this to set and check if dynamic time sequence is enabled 230
tflite::EmbeddingLookupSparseOptionsBuilder . . . 232
nntrainer::props::Epsilon
Epsilon property, this is used to avoid divide by zero . . . 232

tflite::EqualOptionsBuilder . . . 235
nntrainer::exception::ErrorNotification< Err, >
Error Notification class, error is thrown when the class is destroyed. DO NOT use this outside as this contains throwing destructor . . . 235
tflite::ExpandDimsOptionsBuilder . . . 237
tflite::ExpOptionsBuilder . . . 238
nntrainer::Exporter
Exporter class helps to exports the node information in a predefined way. because each method will require complete different methods, this class exploits visitor pattern to make a custom defined saving method . . . 238
External
External Loop Info for broadcasted iteration. Please refer to DISABLED_private_external_loop_n in unittest_nntrainer_tensor . . . 241
tflite::FakeQuantOptionsBuilder . . . 242
nntrainer::props::FilePath
Props containing file path value . . . 243
tflite::FillOptionsBuilder . . . 246
nntrainer::props::FilterSize
FilterSize property, filter size is used to measure how many filters are there . . . 247
nntrainer::FlatBufferInterpreter
Flatbuffer graph interpreter class . . . 249
nntrainer::FlatBufferOpNode
FlatBufferOpNode class . . . 251
tflite::FLATBUFFERS_FINAL_CLASS . . . 259
nntrainer::props::Flatten
Flatten property, true if needs flatten layer afterwards . . . 270
nntrainer::FlattenRealizer
Graph realizer class . . . 272
nntrainer::props::FlipDirection
FlipDirection Enumeration Information . . . 274
nntrainer::props::FlipDirectionInfo
Enumeration of flip direction . . . 275
tflite::FloorDivOptionsBuilder . . . 276
tflite::FloorModOptionsBuilder . . . 276
tflite::FullyConnectedOptionsBuilder . . . 277
nntrainer::FuncDataProducer
FuncDataProducer which contains a callback and returns back . . . 278
tflite::GatherNdOptionsBuilder . . . 281
tflite::GatherOptionsBuilder . . . 282
nntrainer::profile::GenericProfileListener
Generic Profiler Listener . . . 282
nntrainer::props::GenericShape
Generic shape property which saves a single tensor shape (practically, std::array<GenericShape> is used) . . . 285
nntrainer::GraphCompiler
Pure virtual class for the Graph Compiler . . . 288
nntrainer::GraphInterpreter
Pure virtual class for the Graph Interpreter . . . 289
nntrainer::GraphNode . . . 292
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >
Iterator for GraphNode which return const std::shared_ptr<LayerNodeType> object upon realize 296
nntrainer::GraphNodeReverseIterator< T_iterator >
Reverse Iterator for GraphNode which return LayerNode object upon realize . . . 304
nntrainer::GraphRealizer
Graph realizer class . . . 306
tflite::GreaterEqualOptionsBuilder . . . 308
tflite::GreaterOptionsBuilder . . . 309
tflite::HardSwishOptionsBuilder . . . 309

std::hash< nntrainer::Connection >
Hash specialization for connection . . . 310
nntrainer::props::HiddenStateActivation
HiddenStateActivation Enumeration Information . . . 311
tflite::IfOptionsBuilder . . . 313
nntrainer::props::InDim
InDim property, in dim is the size of vocabulary in the text data . . . 313
nntrainer::IniGraphInterpreter
Ini graph interpreter class . . . 315
nntrainer::IniSection
IniSection class that maps to one ini section . . . 318
nntrainer::props::InitializerInfo
Enumeration of tensor initialization type . . . 329
nntrainer::InitLayerContext . . . 330
nntrainer::IniWrapper
IniWrapper using IniSections . . . 341
nntrainer::props::InputConnection
InputSpec property, this defines connection specification of an input . . . 350
nntrainer::props::InputIsSequence
Identifiers to locate an **input** connection which should be sequenced for the connection . . . 353
nntrainer::InputRealizer
Graph realizer class which remaps input from start -> input layers . . . 354
nntrainer::props::InputShape
Input shape property which saves a single tensor shape (practically, std::array<InputShape> is used) . . . 356
tflite::Int32VectorBuilder . . . 358
nntrainer::props::IntegrateBias
Integrate bias_ih and bias_hh to bias_h to use only 1 bias (Used in rnn variant) . . . 359
nntrainer::Iteration
Iteration class which owns the memory chunk for a single batch . . . 361
nntrainer::props::Iteration
Iteration props . . . 366
nntrainer::IterationQueue
Iteration queue that owns the buffer for input / labels . . . 368
nntrainer::props::KernelSize
KernelSize property, kernel size is used to measure the filter size . . . 372
tflite::L2NormOptionsBuilder . . . 373
nntrainer::props::LabelLayer
Label Layer name property which saves a single connection (practically, std::vector<LabelLayers> is used) . . . 374
Layer
Base class for all layers . . . 377
Layer
Class for Layer context . . . 377
Layer
Class for Layer context . . . 377
nntrainer::LayerNode
Layer node class for the graph . . . 377
tflite::LeakyReluOptionsBuilder . . . 411
nntrainer::props::LearningRate
Learning Rate props . . . 411
tflite::LessEqualOptionsBuilder . . . 413
tflite::LessOptionsBuilder . . . 413
tflite::LocalResponseNormalizationOptionsBuilder . . . 414
tflite::LogicalAndOptionsBuilder . . . 415
tflite::LogicalNotOptionsBuilder . . . 415
tflite::LogicalOrOptionsBuilder . . . 416
tflite::LogSoftmaxOptionsBuilder . . . 416

nntrainer::props::Loss
Loss property, this defines loss specification of layer . . . 417
nntrainer::LossRealizer
Graph realizer class which removes loss layer from the graph . . . 420
tflite::LSHProjectionOptionsBuilder . . . 423
tflite::LSTMOptionsBuilder . . . 423
tflite::MatrixDiagOptionsBuilder . . . 424
tflite::MatrixSetDiagOptionsBuilder . . . 424
tflite::MaximumMinimumOptionsBuilder . . . 425
nntrainer::props::MaxTimestep
Maximum timestep property, timestep is used to identify for the maximum time unroll possible for lstm/gru/rnn layer . . . 426
nntrainer::MemoryData< T >
MemoryData Class . . . 427
nntrainer::MemoryPlanner
Memory Planner provides the plan/strategy to allocate the memory . . . 429
nntrainer::MemoryPool
Memory Pool provides a common pool for all the tensor memory . . . 431
nntrainer::MemoryRequest
Memory Request data structure clubbing all the requests . . . 438
tflite::MetadataBuilder . . . 440
tflite::MirrorPadOptionsBuilder . . . 441
tflite::ModelBuilder . . . 441
nntrainer::props::MoL_K
K property, K is the size of the three projections in MoL attention . . . 442
nntrainer::props::Momentum
Momentum property, moving average in batch normalization layer . . . 444
tflite::MulOptionsBuilder . . . 446
nntrainer::MultioutRealizer
Add multiout layer when a certain input is referenced multiple times . . . 447
nntrainer::props::Name
Name property, name is an identifier of an object . . . 449
tflite::NegOptionsBuilder . . . 452
tflite::NonMaxSuppressionV4OptionsBuilder . . . 453
tflite::NonMaxSuppressionV5OptionsBuilder . . . 453
nntrainer::props::Normalization
Normalization property, normalize the input to be in range [0, 1] if true . . . 454
nntrainer::exception::not_supported
Derived class of invalid argument to represent specific functionality not supported . . . 456
tflite::NotEqualOptionsBuilder . . . 457
nntrainer::props::NumClass
Number of class . . . 457
nntrainer::props::NumHeads
NumHeads property, NumHeads is number of head in multi head attention . . . 459
tflite::OneHotOptionsBuilder . . . 461
Op
Class for Optimizer context . . . 462
tflite::OperatorBuilder . . . 462
tflite::OperatorCodeBuilder . . . 463
nntrainer::OptimizedV1Planner
Optimized V1 Memory Planner provides the optimized plan for memory layout . . . 464
nntrainer::OptimizedV2Planner
Optimized V2 Memory Planner provides the optimized plan for memory layout . . . 466
nntrainer::OptimizedV3Planner
Optimized V3 Memory Planner provides the optimized plan for memory layout . . . 469
nntrainer::props::OutDim
OutDim property, out dim is the size of the vector space in which words will be embedded . . . 471

nntrainer::props::OutputLayer
Output Layer name property which saves a single connection (practically, std::vector<Input↩ Layers> is used) . . . 473
nntrainer::props::OutputShape
OutputShape property, output shape of multi head attention . . . 475
tflite::PackOptionsBuilder . . . 477
nntrainer::props::Padding1D
Padding1D property, this is used to calculate padding2D . . . 478
nntrainer::props::Padding2D
Padding2D property, this is used to calculate padding2D . . . 481
nntrainer::props::Padding_
Unsigned integer property, internally used to parse padding values . . . 484
tflite::PadOptionsBuilder . . . 485
tflite::PadV2OptionsBuilder . . . 485
nntrainer::ParallelBatch
ParallelBatch class to parallelize along batch direction . . . 486
nntrainer::exception::permission_denied
Derived class of invalid argument to represent permission is denied . . . 488
nntrainer::props::PermuteDims
PermuteDims property, direction property describes the axis to be transposed. to be used with array . . . 489
nntrainer::PermuteLayer
Permute layer to transpose a tensor . . . 492
nntrainer::internal::PluggedLayer
PluggedLayer to wrap a layer from shared object file . . . 496
nntrainer::internal::PluggedOptimizer
Plugged optimizer class . . . 501
tflite::Pool2DOptionsBuilder . . . 506
nntrainer::props::PoolingType
Pooling Type Enumeration Information . . . 506
nntrainer::props::PoolingTypeInfo
Enumeration of pooling type . . . 507
nntrainer::props::PoolSize
PoolSize property, pool size is used to measure the pooling size . . . 509
tflite::PowOptionsBuilder . . . 511
nntrainer::PreprocessL2NormLayer
Layer class that l2normalizes a feature vector . . . 512
nntrainer::PreviousInputRealizer
Add default inputs if input connection is empty. if a node is identified as input with *identified_input* by user or a node has input_shape property, adding input behavior is skipped . . . 516
nntrainer::profile::ProfileEventData
Data for each profile event . . . 518
nntrainer::profile::ProfileListener
Generic profile listener class to attach to a profiler, this can be inherited to create a custom profile listener . . . 519
nntrainer::profile::Profiler
Profiler object . . . 522
nntrainer::props::ProjectedKeyDim
ProjectedKeyDim property, projected key dim per head in multi head attention . . . 527
nntrainer::props::ProjectedValueDim
ProjectedValueDim property, projected value dim per head in multi head attention . . . 529
nntrainer::PropsBufferSize
Props containing buffer size value . . . 531
nntrainer::PropsMax
Props containing max value . . . 533
nntrainer::PropsMin
Props containing min value . . . 535

nntrainer::PropsNNSModelPath
NNSModelPath property . . . 537
nntrainer::PropsNumSamples
Props containing number of samples A random data producer has theoretical size. number of samples is used to set theoretical size of the random data producer's data size . . . 539
nntrainer::PropsTflModelPath
TflModelPath property . . . 541
nntrainer::props::PropsUserData
User data props . . . 544
tflite::QuantizationDetailsTraits< T > . . . 545
tflite::QuantizationDetailsTraits< tflite::CustomQuantization > . . . 545
tflite::QuantizationParametersBuilder . . . 546
tflite::QuantizeOptionsBuilder . . . 546
nntrainer::RandomDataOneHotProducer
RandomDataProducer which generates a onehot vector as a label . . . 547
nntrainer::props::RandomTranslate
TranslationFactor property, this defines how far the image is translated . . . 551
tflite::RangeOptionsBuilder . . . 553
tflite::RankOptionsBuilder . . . 554
nntrainer::RawFileDataProducer
RawFileDataProducer which contains a callback and returns back . . . 554
nntrainer::props::RecurrentActivation
RecurrentActivation Enumeration Information . . . 559
nntrainer::props::RecurrentInput
Property for recurrent inputs . . . 560
nntrainer::props::RecurrentOutput
Property for recurrent outputs . . . 562
nntrainer::RecurrentRealizer
Recurrent Realizer which unrolls graph from given graph representation . . . 564
nntrainer::props::ReduceDimension
ReduceDimension property, dimension along which to reduce the input . . . 568
tflite::ReducerOptionsBuilder . . . 569
nntrainer::props::RegularizerInfo
Enumeration of tensor regularization type . . . 569
nntrainer::RemapRealizer
Graph realizer class which remaps identifiers inside the graph representation, remap_function will be applied for all the layers identifier visible . . . 571
nntrainer::props::ResetAfter
ResetAfter property, apply reset gate after matrix multiplication if this property is true. Apply before the multiplication if false. Used in gru, grucell . . . 573
tflite::ReshapeOptionsBuilder . . . 575
tflite::ResizeBilinearOptionsBuilder . . . 576
tflite::ResizeNearestNeighborOptionsBuilder . . . 577
nntrainer::props::ReturnAttentionWeight
ReturnAttentionWeight, return attention weight . . . 577
nntrainer::props::ReturnAttentionWeightInfo
Enumeration of return attention weight . . . 579
nntrainer::props::ReturnSequences
Return sequence property, used to check whether return only the last output. Return last output if true . . . 581
tflite::ReverseSequenceOptionsBuilder . . . 582
tflite::ReverseV2OptionsBuilder . . . 583
tflite::RNNOptionsBuilder . . . 583
nntrainer::RunLayerContext . . . 584
nntrainer::RunOptimizerContext . . . 610
nntrainer::Sample
Sample class which views the memory for a single sample . . . 614
tflite::ScatterNdOptionsBuilder . . . 616

nntrainer::ScopedView< T >
A view container that calls a callback on destruct . . . 616
tflite::SegmentSumOptionsBuilder . . . 619
tflite::SelectOptionsBuilder . . . 619
tflite::SelectV2OptionsBuilder . . . 620
tflite::SequenceRNNOptionsBuilder . . . 621
tflite::ShapeOptionsBuilder . . . 621
nntrainer::props::SharedFrom
Properties for shared from . . . 622
tflite::SignatureDefBuilder . . . 624
tflite::SkipGramOptionsBuilder . . . 624
tflite::SliceOptionsBuilder . . . 625
nntrainer::SliceRealizer
Graph realizer class which slice graph representation . . . 626
tflite::SoftmaxOptionsBuilder . . . 628
tflite::SpaceToBatchNDOptionsBuilder . . . 629
tflite::SpaceToDepthOptionsBuilder . . . 630
tflite::SparseIndexVectorTraits< T > . . . 630
tflite::SparseIndexVectorTraits< tflite::Int32Vector > . . . 631
tflite::SparseIndexVectorTraits< tflite::Uint16Vector > . . . 631
tflite::SparseIndexVectorTraits< tflite::Uint8Vector > . . . 631
tflite::SparseToDenseOptionsBuilder . . . 632
tflite::SparsityParametersBuilder . . . 632
nntrainer::props::SplitDimension
SplitDimension property, dimension along which to split the input . . . 633
nntrainer::props::SplitNumber
Split number property, split number indicates how many numbers of outs are generated by splitting the input dimension . . . 635
tflite::SplitOptionsBuilder . . . 637
tflite::SplitVOptionsBuilder . . . 637
tflite::SquaredDifferenceOptionsBuilder . . . 638
tflite::SquareOptionsBuilder . . . 638
tflite::SqueezeOptionsBuilder . . . 639
nntrainer::SrcSharedTensor
Source of the shared tensor . . . 639
nntrainer::props::Standardization
Standardization property, standardization standardize the input to be mean 0 and std 1 if true . 640
nntrainer::props::Stride
Stride property, stride is used to measure how much it will be slide the filter . . . 642
tflite::StridedSliceOptionsBuilder . . . 644
tflite::SubGraphBuilder . . . 644
tflite::SubOptionsBuilder . . . 645
tflite::SVDFOptionsBuilder . . . 645
nntrainer::SwapDevice
A device used to storing data with long access time . . . 646
nntrainer::props::TargetShape
Target shape property which saves a single tensor shape (practically, std::array<TargetShape> is used) . . . 650
nntrainer::Task
Task class . . . 652
nntrainer::TaskAsync< T >
Async task class . . . 656
nntrainer::TaskExecutor
Task executor class . . . 659
tflite::TensorBuilder . . . 665
tflite::TensorMapBuilder . . . 666
nntrainer::TensorSpecV2
Tensor Specification which describes how this tensor should be allocated and managed . . . 666

nntrainer::TfliteInterpreter
Tflite graph interpreter class . . . 670
nntrainer::TfOpNode
Tensorflow operational node representation. This class contains, information to build operation flatbuffer . . . 673
tflite::TileOptionsBuilder . . . 684
nntrainer::props::Timestep
Timestep property, timestep is used to identify for which timestep should the lstm/gru/rnn layer do the operation for . . . 685
tflite::TopKV2OptionsBuilder . . . 686
nntrainer::props::Trainable
Trainable property, use this to set and check how if certain layer is trainable . . . 687
tflite::TransposeConvOptionsBuilder . . . 689
tflite::TransposeOptionsBuilder . . . 689
tflite::Uint16VectorBuilder . . . 690
tflite::Uint8VectorBuilder . . . 690
tflite::UnidirectionalSequenceLSTMOptionsBuilder . . . 691
tflite::UniqueOptionsBuilder . . . 692
nntrainer::props::Unit
Unit property, unit is used to measure how many weights are there . . . 692
tflite::UnpackOptionsBuilder . . . 694
nntrainer::props::UnrollFor
Property check unroll_for . . . 695
nntrainer::Var_Grad
Variable with Gradient, and its corresponding need_gradient property . . . 696
nntrainer::VarGradSpecV2
Variable + gradient specification . . . 709
nntrainer::ViewQueue< T >
Thread Safe Queue implementation dedicated for the non-owing pointer . . . 712
nntrainer::Weight
Weight extends over Var_Grad with regularization & optimizer updates . . . 715
nntrainer::props::WeightDecay
WeightDecay property, this defines how much to decay the weight . . . 725
nntrainer::props::WeightInitializer
WeightInitializer Initialization Enumeration Information . . . 727
nntrainer::props::WeightRegularizer
WeightRegularizer Regularization Enumeration Information . . . 728
nntrainer::props::WeightRegularizerConstant
WeightRegularizerConstant property, this defines how much regularize the weight . . . 730
nntrainer::WeightSpecV2
Weight specification . . . 732
nntrainer::WGradMemoryRequest
Memory Request data structure clubbing for the weight gradient requests . . . 734
tflite::WhereOptionsBuilder . . . 734
tflite::WhileOptionsBuilder . . . 735
nntrainer::props::ZeroIdxMask
Zero idx mask property for embedding where the value of embedding will be zero . . . 736
tflite::ZerosLikeOptionsBuilder . . . 737

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

app_context.cpp
: This file contains app context related functions and classes that manages the global configuration of the current environment . . . 739

app_context.h
: This file contains app context related functions and classes that manages the global configuration of the current environment . . . 741

delegate.h
: This is Delegate Class for the Neural Network . . . 800

nntrainer_error.h
: NNTrainer Error Codes . . . 906

nntrainer_log.h
: NNTrainer Logger. Log Util for NNTrainer . . . 909

nntrainer_logger.cpp
: NNTrainer Logger This allows to logging nntrainer logs . . . 912

nntrainer_logger.h
: NNTrainer Logger This allows to logging nntrainer logs . . . 913

compiler/activation_realizer.cpp
: NNTrainer graph realizer which realizes activation!=none to actual activation node . . . 743

compiler/activation_realizer.h
: NNTrainer graph realizer which realizes activation!=none to actual activation node . . . 744

compiler/bn_realizer.cpp
: NNTrainer graph realizer which remove batch normalization layer for inference . . . 746

compiler/bn_realizer.h
: NNTrainer graph realizer which remove batch normalization layer for inference . . . 747

compiler/compiler.h
: NNTrainer compiler that reads to generate optimized graph . . . 748

compiler/compiler_fwd.h
: NNTrainer graph compiler related forward declarations . . . 749

compiler/flatbuffer_interpreter.cpp
: NNTrainer ∗.flatbuffer Interpreter . . . 750

compiler/flatbuffer_interpreter.h
: NNTrainer flatbuffer Interpreter . . . 751

compiler/**flatbuffer_opnode.cpp** . . . **??**

compiler/flatbuffer_opnode.h
: NNTrainer flatbuffer opnode . . . 752

compiler/flatten_realizer.cpp
NNTrainer graph realizer which realizes flatten=true to actual node . . . 754
compiler/flatten_realizer.h
NNTrainer graph realizer which realizes flatten=true to actual node . . . 755
compiler/ini_interpreter.cpp
NNTrainer Ini Interpreter (partly moved from model_loader.c) . . . 756
compiler/ini_interpreter.h
NNTrainer Ini Interpreter . . . 757
compiler/input_realizer.cpp
NNTrainer graph realizer which remaps input to the external graph . . . 759
compiler/input_realizer.h
NNTrainer graph realizer which remaps input to the external graph . . . 760
compiler/interpreter.h
NNTrainer interpreter that reads and generates a graphRepresentation from a file . . . 761
compiler/**loss_realizer.cpp** . . . **??**
compiler/loss_realizer.h
NNTrainer graph realizer which remove loss layer for inference . . . 763
compiler/**multiout_realizer.cpp** . . . **??**
compiler/multiout_realizer.h
NNTrainer graph realizer which realizes multiout to actual node . . . 765
compiler/previous_input_realizer.cpp
NNTrainer graph realizer which connects input to previous one if empty . . . 767
compiler/previous_input_realizer.h
NNTrainer graph realizer which connects input to previous one if empty . . . 768
compiler/realizer.h
NNTrainer graph realizer which preprocess graph representation as a lowering process of compile . . . 769
compiler/**recurrent_realizer.cpp** . . . **??**
compiler/recurrent_realizer.h
NNTrainer graph realizer to create unrolled graph from a graph realizer . . . 770
compiler/**remap_realizer.cpp** . . . **??**
compiler/remap_realizer.h
NNTrainer graph realizer which realizes identifier to a new identifier . . . 772
compiler/slice_realizer.cpp
NNTrainer graph realizer which slice the graph representation . . . 774
compiler/slice_realizer.h
NNTrainer graph realizer which slice the graph representation . . . 775
compiler/**tf_schema_generated.h** . . . **??**
compiler/tflite_interpreter.cpp
NNTrainer ∗.tflite Interpreter . . . 776
compiler/tflite_interpreter.h
NNTrainer ∗.tflite Interpreter . . . 777
compiler/tflite_opnode.cpp
Tflite opnode which has information to convert to tflite file . . . 778
compiler/tflite_opnode.h
Tflite opnode which has information to convert to tflite file . . . 779
dataset/data_iteration.cpp
This file contains iteration and sample class . . . 781
dataset/data_iteration.h
This file contains iteration and sample class . . . 782
dataset/data_producer.h
This file contains data producer interface . . . 783
dataset/databuffer.cpp
This is buffer object to handle big data . . . 784
dataset/databuffer.h
This is buffer object to handle big data . . . 785
dataset/databuffer_factory.cpp
This is the databuffer factory . . . 786

dataset/databuffer_factory.h
This is the layer factory . . . 787
dataset/**dir_data_producers.cpp** . . . **??**
dataset/dir_data_producers.h
This file contains dir data producers, reading from the files in directory . . . 788
dataset/func_data_producer.cpp
This file contains various data producers from a callback . . . 790
dataset/func_data_producer.h
This file contains various data producers from a callback . . . 791
dataset/iteration_queue.cpp
This file contains thread safe queue . . . 792
dataset/iteration_queue.h
This file contains thread safe queue for a single iteration . . . 793
dataset/random_data_producers.cpp
This file contains various random data producers . . . 795
dataset/random_data_producers.h
This file contains various random data producers . . . 796
dataset/raw_file_data_producer.cpp
This file contains raw file data producers, reading from a file . . . 797
dataset/raw_file_data_producer.h
This file contains raw file data producers, reading from a file . . . 798
graph/connection.cpp
Connection class and related utility functions . . . 801
graph/connection.h
Connection class and related utility functions . . . 802
graph/**graph_core.cpp** . . . **??**
graph/**graph_core.h** . . . **??**
graph/graph_node.h
This is the graph node interface for c++ API . . . 803
graph/**network_graph.cpp** . . . **??**
graph/network_graph.h
This is Graph Core Class for Neural Network . . . 804
layers/acti_func.cpp
This is Activation Layer Class for Neural Network . . . 806
layers/**acti_func.h** . . . **??**
layers/activation_layer.cpp
This is Activation Layer Class for Neural Network . . . 807
layers/activation_layer.h
This is Activation Layer Class for Neural Network . . . 808
layers/addition_layer.cpp
This is Addition Layer Class for Neural Network . . . 809
layers/addition_layer.h
This is Addition Layer Class for Neural Network . . . 810
layers/attention_layer.cpp
This is Attention Layer Class for Neural Network . . . 811
layers/attention_layer.h
This is Attention Layer Class for Neural Network . . . 812
layers/bn_layer.cpp
This is Batch Normalization Layer Class for Neural Network . . . 813
layers/bn_layer.h
This is Batch Normalization Layer Class of Neural Network . . . 814
layers/centroid_knn.cpp
This file contains the simple nearest neighbor layer . . . 815
layers/centroid_knn.h . . . 816
layers/common_properties.cpp
This file contains implementation of common properties widely used across layers . . . 817
layers/common_properties.h
This file contains list of common properties widely used across layers . . . 818

layers/concat_layer.cpp
This is Concat Layer Class for Neural Network . . . 823
layers/concat_layer.h
This is Concat Layer Class for Neural Network . . . 824
layers/**conv1d_layer.cpp** . . . **??**
layers/conv1d_layer.h
This is Convolution 1D Layer Class for Neural Network . . . 824
layers/**conv2d_layer.cpp** . . . **??**
layers/conv2d_layer.h
This is Convolution Layer Class for Neural Network . . . 825
layers/**dropout.cpp** . . . **??**
layers/dropout.h
This is DropOut Layer Class for Neural Network . . . 826
layers/embedding.cpp
This is Embedding Layer Class of Neural Network . . . 827
layers/embedding.h
This is Embedding Layer Class of Neural Network . . . 828
layers/**entropy_layer.h** . . . **??**
layers/fc_layer.cpp
This is Fully Connected Layer Class for Neural Network . . . 829
layers/fc_layer.h
This is Fully Connected Layer Class of Neural Network . . . 830
layers/flatten_layer.cpp
This is Flatten Layer Class for Neural Network . . . 831
layers/flatten_layer.h
This is Flatten Layer Class for Neural Network . . . 832
layers/gru.cpp
This is Gated Recurrent Unit Layer Class of Neural Network . . . 833
layers/gru.h
This is Gated Recurrent Unit Layer Class of Neural Network . . . 834
layers/grucell.cpp
This is Gated Recurrent Unit Cell Layer Class of Neural Network . . . 835
layers/grucell.h
This is Gated Recurrent Unit Cell Layer Class of Neural Network . . . 836
layers/**identity_layer.cpp** . . . **??**
layers/**identity_layer.h** . . . **??**
layers/input_layer.cpp
This is Input Layer Class for Neural Network . . . 837
layers/input_layer.h
This is Input Layer Class of Neural Network . . . 838
layers/layer_context.cpp
This is the layer context for each layer . . . 839
layers/layer_context.h
This is the layer context for each layer . . . 840
layers/layer_devel.h
This is Layer classes of Neural Network . . . 841
layers/**layer_impl.cpp** . . . **??**
layers/layer_impl.h
This is base layer implementation class . . . 842
layers/layer_node.cpp
This is the layer node for network graph . . . 844
layers/layer_node.h
This is the layer node for network graph . . . 846
layers/layer_normalization_layer.cpp
This is Layer Normalization Layer Class for Neural Network . . . 848
layers/layer_normalization_layer.h
This is Layer Normalization Layer Class for Neural Network . . . 849

layers/lstm.cpp
This is Long Short-Term Memory Layer Class of Neural Network . . . 858
layers/lstm.h
This is Long Short-Term Memory Layer Class of Neural Network . . . 859
layers/lstmcell.cpp
This is LSTMCell Layer Class of Neural Network . . . 860
layers/lstmcell.h
This is LSTMCell Layer Class of Neural Network . . . 861
layers/lstmcell_core.cpp
This is lstm core class . . . 862
layers/lstmcell_core.h
This is lstm core class . . . 863
layers/mol_attention_layer.cpp
This is MoL Attention Layer Class for Neural Network . . . 864
layers/mol_attention_layer.h
This is MoL Attention Layer Class for Neural Network . . . 865
layers/multi_head_attention_layer.cpp
This is MultiHeadAttention Layer Class for Neural Network . . . 866
layers/multi_head_attention_layer.h
This is MultiHeadAttention Layer Class for Neural Network . . . 867
layers/multiout_layer.cpp
This is Multi Output Layer Class for Neural Network . . . 868
layers/multiout_layer.h
This is Multi Output Layer Class for Neural Network . . . 869
layers/nnstreamer_layer.cpp
This is class to encapsulate nnstreamer as a layer of Neural Network . . . 869
layers/nnstreamer_layer.h
This is class to encapsulate nnstreamer as a layer of Neural Network . . . 871
layers/permute_layer.cpp
Permute layer to support transpose . . . 872
layers/permute_layer.h
Permute layer to support transpose . . . 873
layers/plugged_layer.h
This file contains a wrapper for a plugged layer, INTERNAL USE ONLY . . . 874
layers/pooling2d_layer.cpp
This is 2 Dimensional Pooling Layer Class for Neural Network . . . 875
layers/pooling2d_layer.h
This is 2 Dimensional Pooling Layer Class for Neural Network . . . 876
layers/positional_encoding_layer.cpp
This file contains the positional encoding layer in transformer . . . 877
layers/positional_encoding_layer.h
This file contains the positional encoding layer in transformer . . . 878
layers/preprocess_flip_layer.cpp
This is Preprocess Random Flip Layer Class for Neural Network . . . 879
layers/preprocess_flip_layer.h
This is Preprocess Random Flip Layer Class for Neural Network . . . 880
layers/preprocess_l2norm_layer.cpp
This file contains the simple l2norm layer which normalizes the given feature . . . 881
layers/preprocess_l2norm_layer.h
This file contains the simple l2norm layer which normalizes the given feature . . . 882
layers/**preprocess_translate_layer.cpp** . . . **??**
layers/**preprocess_translate_layer.h** . . . **??**
layers/reduce_mean_layer.cpp
This is Reduce Mean Layer Class for Neural Network . . . 883
layers/reduce_mean_layer.h
This is Reduce Mean Layer Class for Neural Network . . . 884
layers/**reshape_layer.cpp** . . . **??**

layers/reshape_layer.h
This is Reshape Layer Class for Neural Network . . . 885
layers/rnn.cpp
This is Recurrent Layer Class of Neural Network . . . 886
layers/rnn.h
This is Recurrent Layer Class of Neural Network . . . 887
layers/rnncell.cpp
This is Recurrent Cell Layer Class of Neural Network . . . 887
layers/rnncell.h
This is Recurrent Layer Cell Class of Neural Network . . . 889
layers/split_layer.cpp
This is Split Layer Class for Neural Network . . . 889
layers/split_layer.h
This is Split Layer Class for Neural Network . . . 890
layers/tflite_layer.cpp
This is class to encapsulate tflite as a layer of Neural Network . . . 891
layers/tflite_layer.h
This is class to encapsulate tflite as a layer of Neural Network . . . 892
layers/time_dist.cpp
This is Time Distributed Layer Class of Neural Network . . . 893
layers/time_dist.h
This is Time Distributed Layer Class of Neural Network . . . 894
layers/zoneout_lstmcell.cpp
This is ZoneoutLSTMCell Layer Class of Neural Network . . . 895
layers/zoneout_lstmcell.h
This is ZoneoutLSTMCell Layer Class of Neural Network . . . 896
layers/loss/constant_derivative_loss_layer.cpp
This patch contains constant derivative loss implementation . . . 849
layers/loss/**constant_derivative_loss_layer.h** . . . **??**
layers/loss/cross_entropy_loss_layer.h
This is Cross Entropy Loss Layer Class of Neural Network . . . 850
layers/loss/cross_entropy_sigmoid_loss_layer.cpp
This is MSE Loss Layer Class of Neural Network . . . 851
layers/loss/cross_entropy_sigmoid_loss_layer.h
This is Cross Entropy Sigmoid with Sigmoid Loss Layer Class of Neural Network . . . 852
layers/loss/**cross_entropy_softmax_loss_layer.cpp** . . . **??**
layers/loss/**cross_entropy_softmax_loss_layer.h** . . . **??**
layers/loss/kld_loss_layer.cpp
KLD (Kullback-Leibler Divergence) loss implementation . . . 853
layers/loss/kld_loss_layer.h
KLD (Kullback-Leibler Divergence) loss implementation . . . 854
layers/loss/loss_layer.cpp
This is Loss Layer Class for Neural Network . . . 855
layers/loss/loss_layer.h
This is Loss Layer Class of Neural Network . . . 856
layers/loss/mse_loss_layer.cpp
This is MSE Loss Layer Class of Neural Network . . . 857
layers/loss/mse_loss_layer.h
This is MSE Loss Layer Class of Neural Network . . . 858
models/dynamic_training_optimization.cpp
This is Dynamic Training Optimization for Neural Network . . . 896
models/dynamic_training_optimization.h
This is Dynamic Training Optimization for Neural Network . . . 898
models/execution_mode.h
This is mode of executions . . . 899
models/model_common_properties.cpp
This file contains common properties for model . . . 899

models/model_common_properties.h
This file contains common properties for model . . . 900
models/model_loader.cpp
This is model loader class for the Neural Network . . . 901
models/model_loader.h
This is model loader class for the Neural Network . . . 903
models/neuralnet.cpp
This is Neural Network Class . . . 904
models/neuralnet.h
This is Neural Network Class . . . 905
optimizers/adam.cpp
This is the Adam optimizer . . . 914
optimizers/adam.h
This is the Adam optimizer . . . 915
optimizers/lr_scheduler.h
This is Learning Rate Scheduler interface class . . . 915
optimizers/lr_scheduler_constant.cpp
This is Constant Learning Rate Scheduler class . . . 916
optimizers/lr_scheduler_constant.h
This is Constant Learning Rate Scheduler class . . . 917
optimizers/lr_scheduler_exponential.cpp
This is Exponential Learning Rate Scheduler class . . . 918
optimizers/lr_scheduler_exponential.h
This is Exponential Learning Rate Scheduler class . . . 919
optimizers/lr_scheduler_step.cpp
This is Step Learning Rate Scheduler class . . . 919
optimizers/lr_scheduler_step.h
This is Step Learning Rate Scheduler class . . . 921
optimizers/**optimizer_context.cpp** . . . **??**
optimizers/optimizer_context.h
This is the layer context for each layer . . . 922
optimizers/optimizer_devel.cpp
This is Optimizer internal interface class . . . 923
optimizers/optimizer_devel.h
This is Optimizer internal interface class . . . 924
optimizers/optimizer_wrapped.cpp
This is Optimizer Wrapped interface class . . . 925
optimizers/optimizer_wrapped.h
This is Optimizer Wrapped interface class . . . 926
optimizers/plugged_optimizer.h
This file contains a wrapper for a plugged optimizer, INTERNAL USE ONLY . . . 927
optimizers/sgd.cpp
This is the SGD optimizer . . . 929
optimizers/sgd.h
This is the SGD optimizer . . . 930
tensor/basic_planner.cpp
This is Naive Memory Planner . . . 931
tensor/basic_planner.h
This is Naive Memory Planner . . . 932
tensor/blas_interface.cpp
This is dummy header for blas support . . . 933
tensor/blas_interface.h
This is dummy header for blas support . . . 935
tensor/cache_elem.cpp
Cache elem class . . . 936
tensor/cache_elem.h
Cache elem class . . . 937

tensor/cache_loader.cpp
Cache loader class . . . 938
tensor/cache_loader.h
Cache loader class . . . 939
tensor/cache_pool.cpp
Cache pool class inherited from memory pool . . . 941
tensor/cache_pool.h
Cache pool class inherited from memory pool . . . 942
tensor/lazy_tensor.cpp
A lazy evaluation calculator for tensors . . . 943
tensor/lazy_tensor.h
A lazy evaluation calculator for tensors . . . 944
tensor/manager.cpp
This is NNtrainer manager for all weights, i/o and intermediate tensors . . . 945
tensor/manager.h
This is NNtrainer manager for all weights, i/o and intermediate tensors . . . 946
tensor/memory_data.h
MemoryData class . . . 948
tensor/memory_planner.h
This is interface for the Memory Planner . . . 949
tensor/memory_pool.cpp
This is Memory Pool Class . . . 950
tensor/memory_pool.h
This is Memory Pool Class . . . 951
tensor/optimized_v1_planner.cpp
This is Optimized V1 Memory Planner . . . 953
tensor/**optimized_v1_planner.h** . . . **??**
tensor/optimized_v2_planner.cpp
This is Optimized V2 Memory Planner . . . 954
tensor/**optimized_v2_planner.h** . . . **??**
tensor/optimized_v3_planner.cpp
This is Optimized V3 Memory Planner . . . 955
tensor/**optimized_v3_planner.h** . . . **??**
tensor/swap_device.cpp
Swap device class implementation . . . 956
tensor/**swap_device.h** . . . **??**
tensor/task.h
Task class . . . 958
tensor/task_executor.cpp
Task executor class . . . 959
tensor/task_executor.h
Task executor class . . . 960
tensor/tensor.cpp
This is Tensor class for calculation . . . 962
tensor/tensor.h
This is Tensor class for calculation . . . 964
tensor/tensor_dim.cpp
This is Tensor Dimension Class . . . 965
tensor/tensor_pool.cpp
This is TensorPool for all requested tensors . . . 966
tensor/tensor_pool.h
This is TensorPool for all requested tensors . . . 967
tensor/tensor_wrap_specs.h
This is specs for various tensor wrappers . . . 968
tensor/var_grad.cpp
This is Var_Grad Class for Neural Network . . . 969
tensor/var_grad.h
This is Var_Grad Class for Neural Network . . . 971

tensor/weight.cpp
This is Weight Class for Neural Network . . . 972
tensor/weight.h
This is Weight Class for Neural Network . . . 973
utils/base_properties.cpp
Convenient property type definition for automated serialization . . . 974
utils/base_properties.h
Convenient property type definition for automated serialization . . . 975
utils/ini_wrapper.cpp
NNTrainer Ini Wrapper helps to save ini . . . 976
utils/ini_wrapper.h
NNTrainer Ini Wrapper helps to save ini . . . 977
utils/nntr_threads.cpp
Thread Management for NNTrainer . . . 978
utils/nntr_threads.h
Thread Management for NNTrainer . . . 979
utils/node_exporter.cpp
NNTrainer Node exporter . . . 980
utils/node_exporter.h
NNTrainer Node exporter . . . 981
utils/profiler.cpp
Profiler related codes to be used to benchmark things . . . 983
utils/profiler.h
Profiler related codes to be used to benchmark things . . . 984
utils/**tracer.cpp** . . . **??**
utils/tracer.h
Trace abstract class . . . 986
utils/util_func.cpp
This is collection of math functions . . . 987
utils/util_func.h
This is collection of math functions . . . 989

Chapter 7

Namespace Documentation

7.1 nntrainer Namespace Reference

Namespaces

- exception

Classes

- class ActivationRealizer

  Graph realizer which realizes activation.
- class AppContext

  App.
- class Backend

  Backend to be used for the operations to use.
- class BasicPlanner

  Basic Memory Planner provides the basic plan for memory layout.
- class BnRealizer

  Graph realizer class which removes batch normalization layer from the graph.
- class CacheElem

  Cache element containing swap address.
- class CacheLoader

  Cache loader from swap device.
- class CachePool

  Cache memory with fixed size utilizing swap device.
- class CentroidKNN

  Centroid KNN layer which takes centroid and do k-nearest neighbor classification.
- class Connection

  RAII class to define a connection.
- class DataProducer

  DataProducer interface used to abstract data provider.
- class DelegateConfig

  Configuration for the delegate.
- class Device

  Device to be used for the operations to run.

- class DirDataProducer

  DirDataProducer which generates a onehot vector as a label.
- class Exporter

  Exporter class helps to exports the node information in a predefined way. because each method will require complete different methods, this class exploits visitor pattern to make a custom defined saving method.
- class FlatBufferInterpreter

  flatbuffer graph interpreter class
- class FlatBufferOpNode

  FlatBufferOpNode class.
- class FlattenRealizer

  Graph realizer class.
- class FuncDataProducer

  FuncDataProducer which contains a callback and returns back.
- class GraphCompiler

  Pure virtual class for the Graph Compiler.
- class GraphInterpreter

  Pure virtual class for the Graph Interpreter.
- class GraphNode
- class GraphNodeIterator

  Iterator for GraphNode which return const std::shared_ptr<LayerNodeType> object upon realize.
- class GraphNodeReverseIterator

  Reverse Iterator for GraphNode which return LayerNode object upon realize.
- class GraphRealizer

  Graph realizer class.
- class IniGraphInterpreter

  ini graph interpreter class
- class IniSection

  IniSection class that maps to one ini section.
- class InitLayerContext
- class IniWrapper

  IniWrapper using IniSections.
- class InputRealizer

  Graph realizer class which remaps input from start -> input layers.
- class Iteration

  Iteration class which owns the memory chunk for a single batch.
- class IterationQueue

  Iteration queue that owns the buffer for input / labels.
- class LayerNode

  layer node class for the graph
- class LossRealizer

  Graph realizer class which removes loss layer from the graph.
- class MemoryData

  MemoryData Class.
- class MemoryPlanner

  Memory Planner provides the plan/strategy to allocate the memory.
- class MemoryPool

  Memory Pool provides a common pool for all the tensor memory.
- struct MemoryRequest

  Memory Request data structure clubbing all the requests.
- class MultioutRealizer

  Add multiout layer when a certain input is referenced multiple times.

- class OptimizedV1Planner

  Optimized V1 Memory Planner provides the optimized plan for memory layout.
- class OptimizedV2Planner

  Optimized V2 Memory Planner provides the optimized plan for memory layout.
- class OptimizedV3Planner

  Optimized V3 Memory Planner provides the optimized plan for memory layout.
- class ParallelBatch

  ParallelBatch class to parallelize along batch direction.
- class PermuteLayer

  Permute layer to transpose a tensor.
- class PreprocessL2NormLayer

  Layer class that l2normalizes a feature vector.
- class PreviousInputRealizer

  Add default inputs if input connection is empty. if a node is identified as input with identified_input by user or a node has input_shape property, adding input behavior is skipped.
- class PropsBufferSize

  Props containing buffer size value.
- class PropsMax

  Props containing max value.
- class PropsMin

  Props containing min value.
- class PropsNNSModelPath

  NNSModelPath property.
- class PropsNumSamples

  Props containing number of samples A random data producer has theoretical size. number of samples is used to set theoretical size of the random data producer's data size.
- class PropsTflModelPath

  TflModelPath property.
- class RandomDataOneHotProducer

  RandomDataProducer which generates a onehot vector as a label.
- class RawFileDataProducer

  RawFileDataProducer which contains a callback and returns back.
- class RecurrentRealizer

  Recurrent Realizer which unrolls graph from given graph representation.
- class RemapRealizer

  Graph realizer class which remaps identifiers inside the graph representation, remap_function will be applied for all the layers identifier visible.
- class RunLayerContext
- class RunOptimizerContext
- class Sample

  Sample class which views the memory for a single sample.
- class ScopedView

  A view container that calls a callback on destruct.
- class SliceRealizer

  Graph realizer class which slice graph representation.
- class SrcSharedTensor

  Source of the shared tensor.
- class SwapDevice

  A device used to storing data with long access time.
- class Task

  task class

- class TaskAsync

  Async task class.
- class TaskExecutor

  task executor class
- struct TensorSpecV2

  Tensor Specification which describes how this tensor should be allocated and managed.
- class TfliteInterpreter

  tflite graph interpreter class
- class TfOpNode

  tensorflow operational node representation. This class contains, information to build operation flatbuffer
- class Var_Grad

  Variable with Gradient, and its corresponding need_gradient property.
- struct VarGradSpecV2

  variable + gradient specification
- class ViewQueue

  Thread Safe Queue implementation dedicated for the non-owing pointer.
- class Weight

  Weight extends over Var_Grad with regularization & optimizer updates.
- struct WeightSpecV2

  weight specification
- struct WGradMemoryRequest

  Memory Request data structure clubbing for the weight gradient requests.

Typedefs

- using **GraphRepresentation** = std::vector< std::shared_ptr< LayerNode > >
- using **ExecutableGraph** = NetworkGraph
- using **datagen_cb** = ml::train::datagen_cb
- template<class LayerNodeType >
  using graph_const_iterator = GraphNodeIterator< LayerNodeType, const std::shared_ptr< GraphNode > >

  Iterators to traverse the graph.
- template<class LayerNodeType >
  using graph_const_reverse_iterator = GraphNodeReverseIterator< GraphNodeIterator< LayerNodeType, const std::shared_ptr< GraphNode > >>

  Iterators to traverse the graph.
- using **MemoryDataValidateCallback** = std::function< void(unsigned int)>
- typedef std::tuple< TensorDim, Tensor::Initializer, WeightRegularizer, float, float, float, bool, const std::string > WeightSpec

  Specification of the Weight as a tensor wrapper.
- typedef std::tuple< TensorDim, Tensor::Initializer, bool, const std::string, TensorLifespan > VarGradSpec

  Specification of the Var_Grad (trainable tensor) as a tensor wrapper.

Enumerations

- enum AttentionParams {
  query = 0, value = 1, **key** = 2, **weights**,
  query_fc_weight, **query_fc_bias**, **key_fc_weight**, **key_fc_bias**,
  value_fc_weight, **value_fc_bias**, **fc_weight**, **fc_bias**,
  projected_query, **projected_key**, **projected_value**, attention_weight,
  dropout_mask, **attention_output** }

- enum **BNParams** {
  mu, **var**, **gamma**, **beta**,
  deviation, **invstd**, **cvar**, **t_reduced**,
  t_full }
- enum **KNNParams** { **map**, **num_samples** }
- enum ActivationType {
  ActivationType::ACT_TANH, ActivationType::ACT_SIGMOID, ActivationType::ACT_RELU, ActivationType::ACT_SWISH,
  ActivationType::ACT_SOFTMAX, ActivationType::ACT_LEAKY_RELU, ActivationType::ACT_NONE,
  ActivationType::ACT_UNKNOWN }

  Enumeration of activation function type.
- enum **EmbeddingParams** { **weight** }
- enum **FCParams** { **weight**, **bias** }
- enum **GRUParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **hidden_state**, **zrg**, **h_prev**,
  dropout_mask }
- enum **GRUCellParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **zrg**, **dropout_mask** }
- enum PrintOption {
  PRINT_INST_INFO = (1 << 0), PRINT_SHAPE_INFO = (1 << 1), PRINT_PROP = (1 << 2),
  PRINT_PROP_META = (1 << 3),
  PRINT_WEIGHTS = (1 << 4), PRINT_METRIC = (1 << 5) }

  Print Options when printing layer info.
- enum InPlace { InPlace::NONE, InPlace::RESTRICTING, InPlace::NON_RESTRICTING }

  Enum class for the various types of inplace modes supported by layer.
- enum **LNParams** {
  gamma, **beta**, **deviation**, **variance**,
  inv_std_dev, **temp_origin_size**, **temp_normalized_size** }
- enum **LSTMParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **hidden_state**, **cell_state**, **ifgo**,
  reverse_weight_ih, **reverse_weight_hh**, **reverse_bias_h**, **reverse_bias_ih**,
  reverse_bias_hh, **reverse_hidden_state**, **reverse_cell_state**, **reverse_ifgo**,
  dropout_mask }
- enum **LSTMCellParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **ifgo**, **dropout_mask** }
- enum MoLAttentionParams {
  query = 0, value = 1, **state** = 2, **mask_len** = 3,
  fc_w, **fc_bias**, **fc_proj_w**, **fc_out**,
  fc_tanh, **fc_proj_out**, **scores**, **prob**,
  prob_left, **prob_right**, **u_neg_div**, **u_pos_div**,
  dstate }
- enum INOUT_INDEX {
  QUERY = 0, **KEY** = 1, **VALUE** = 2, **MASK** = 3,
  OUTPUT = 0, **RETURN_ATTENTION_WEIGHT** = 1 }
- enum AttentionParams {
  query = 0, value = 1, **key** = 2, **weights**,
  query_fc_weight, **query_fc_bias**, **key_fc_weight**, **key_fc_bias**,
  value_fc_weight, **value_fc_bias**, **fc_weight**, **fc_bias**,
  projected_query, **projected_key**, **projected_value**, attention_weight,
  dropout_mask, **attention_output** }
- enum **PositionalEncodingParams** { **positional_encoding** }
- enum **RNNParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **hidden_state**, **dropout_mask** }

- enum **RNNCellParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **dropout_mask** }
- enum **ZoneoutLSTMParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **ifgo**, **hidden_state_zoneout_mask**, **cell_state_zoneout_mask**,
  lstm_cell_state }
- enum ExecutionMode { **TRAIN**, ExecutionMode::INFERENCE, ExecutionMode::VALIDATE }

  class telling the execution mode of the model/operation
- enum **AdamParams** { **wm**, **wv** }
- enum CachePolicy {
  WRITE_BACK = 0b0001, NO_WRITE_BACK = 0b0010, READ_CONSIST = 0b0100, NO_READ_CONSIST = 0b1000,
  ALWAYS_SYNCED, TEMPORAL, FIRST_LAST_SKIP = 0b10000, ITERATION_CONSIST = (FIRST_LA↩ST_SKIP | ALWAYS_SYNCED) }
- enum WeightRegularizer { WeightRegularizer::L2NORM, WeightRegularizer::NONE, WeightRegularizer::UNKNOWN }

  Enumeration of Weight Regularizer.
- enum TensorLifespan {
  TensorLifespan::UNMANAGED = 0b000, TensorLifespan::FORWARD_FUNC_LIFESPAN = 0b001,
  TensorLifespan::CALC_DERIV_LIFESPAN = 0b010, TensorLifespan::CALC_GRAD_LIFESPAN = 0b100,
  TensorLifespan::CALC_AGRAD_LIFESPAN = 0b1000, TensorLifespan::CALC_GRAD_DERIV_LIFESPAN = 0b110, TensorLifespan::CALC_GRAD_DERIV_AGRAD_LIFESPAN = 0b1110, TensorLifespan::FORWARD_GRAD_LIFESPAN = 0b101,
  TensorLifespan::FORWARD_GRAD_AGRAD_LIFESPAN = 0b1101, TensorLifespan::FORWARD_DERIV_LIFESPAN = 0b011, TensorLifespan::BACKWARD_FUNC_LIFESPAN, TensorLifespan::ITERATION_LIFESPAN = 0b1111,
  TensorLifespan::EPOCH_LIFESPAN = 0b11111, TensorLifespan::MAX_LIFESPAN = 0b111111 }

  define the lifespan of the given tensor to reduce peak memory

Functions

- static void fini_global_context_nntrainer (void) __attribute__((destructor))

  finialize global context
- static void add_default_object (AppContext &ac)
- static void **add_extension_object** (AppContext &ac)
- static void **registerer** (AppContext &ac) noexcept
- template<size_t I = 0, typename V , typename... Ts>
  std::enable_if< I==sizeof...(Ts), void >::type parse_properties (V &props, std::tuple< Ts... > &tup)

  base case of iterate_prop, iterate_prop iterates the given tuple
- const template int AppContext::registerFactory< nntrainer::Optimizer > (const FactoryType< nntrainer::↩Optimizer > factory, const std::string &key, const int int_key)

  int AppContext::registerFactory
- const template int AppContext::registerFactory< nntrainer::Layer > (const FactoryType< nntrainer::Layer > factory, const std::string &key, const int int_key)

  int AppContext::registerFactory
- const template int AppContext::registerFactory< ml::train::LearningRateScheduler > (const FactoryType< ml::train::LearningRateScheduler > factory, const std::string &key, const int int_key)

  int AppContext::registerFactory
- static void propagateTimestep (LayerNode *node, unsigned int time_step, unsigned int max_time_step)

  if node is of recurrent type, set time step and max time step
- std::unique_ptr< DataBuffer > createDataBuffer (DatasetType type)

  Factory creator with constructor.

- std::unique_ptr< DataBuffer > createDataBuffer (DatasetType type, const char *file)

  Factory creator with constructor for dataset.
- std::unique_ptr< DataBuffer > createDataBuffer (DatasetType type, datagen_cb cb, void *user_data)

  Factory creator with constructor for dataset.
- static void setInplaceSharedMemoryConfigByLayer (const std::shared_ptr< LayerNode > &lnode, bool &shared_var, bool &shared_grad)

  Set the Inplace Shared Memory Config By Layer object.
- static void grucell_forwarding (const unsigned int unit, const unsigned int batch_size, const bool disable_bias, const bool integrate_bias, const bool reset_after, ActiFunc &acti_func, ActiFunc &recurrent_acti_func, const Tensor &input, const Tensor &prev_hidden_state, Tensor &hidden_state, const Tensor &weight_ih, const Tensor &weight_hh, const Tensor &bias_h, const Tensor &bias_ih, const Tensor &bias_hh, Tensor &zrg)

  gru forwarding
- static void grucell_calcGradient (const unsigned int unit, const unsigned int batch_size, const bool disable_bias, const bool integrate_bias, const bool reset_after, ActiFunc &acti_func, ActiFunc &recurrent_acti_func, const Tensor &input, const Tensor &prev_hidden_state, Tensor &d_prev_hidden_state, const Tensor &d_hidden_state, Tensor &d_weight_ih, const Tensor &weight_hh, Tensor &d_weight_hh, Tensor &d_bias_h, Tensor &d_bias_ih, const Tensor &bias_hh, Tensor &d_bias_hh, const Tensor &zrg, Tensor &d_zrg)

  gru calcGradient
- static void suffixSpec (VarGradSpecV2 &spec, unsigned int idx)

  rename specification
- std::unique_ptr< LayerNode > createLayerNode (const ml::train::LayerType &type, const std::vector< std::string > &properties)

  Layer factory creator with constructor.
- std::unique_ptr< LayerNode > createLayerNode (const std::string &type, const std::vector< std::string > &properties)

  Layer factory creator with constructor.
- std::unique_ptr< LayerNode > createLayerNode (std::unique_ptr< nntrainer::Layer > &&layer, const std::vector< std::string > &properties)

  Layer factory creator with constructor.
- std::ostream & **operator**<< (std::ostream &out, const LayerNode &l)
- static int **nnst_info_to_tensor_dim** (ml_tensors_info_h &out_res, TensorDim &dim)
- static std::string buildTrasposeString (const std::array< props::PermuteDims, 3 > &arr)

  buildTransposeString based on array
- static void **reshape** (Tensor &m)
- void **swap** (NeuralNetwork &lhs, NeuralNetwork &rhs)
- void __nntrainer_log_print (nntrainer_loglevel loglevel, const std::string format_str,...)

  Interface function for C.
- std::unique_ptr< OptimizerWrapped > createOptimizerWrapped (const ml::train::OptimizerType &type, const std::vector< std::string > &properties)

  Optimizer wrapped creator with constructor for optimizer.
- std::unique_ptr< OptimizerWrapped > createOptimizerWrapped (const std::string &type, const std::vector< std::string > &properties)

  Optimizer wrapped creator with constructor for optimizer.
- std::unique_ptr< OptimizerWrapped > createOptimizerWrapped (std::unique_ptr< OptimizerCore > &&opt, const std::vector< std::string > &properties)

  Optimizer wrapped creator with constructor for optimizer.
- static void **saxpy_raw** (const unsigned int N, const float alpha, const float *X, const int incX, float *Y, const int incY)
- static void **sgemv_raw** (CBLAS_ORDER order, CBLAS_TRANSPOSE TransA, const unsigned int M, const unsigned int N, const float alpha, const float *A, const unsigned int lda, const float *X, const int incX, const float beta, float *Y, const int incY)
- static float **sdot_raw** (const unsigned int N, const float *X, const unsigned int incX, const float *Y, const unsigned int incY)

- static void **scopy_raw** (const unsigned int N, const float *X, const int incX, float *Y, const int incY)
- static void **sscal_raw** (const unsigned int N, const float alpha, float *X, const int incX)
- static float **snrm2_raw** (const unsigned int N, const float *X, const int incX)
- static void **sgemm_raw** (CBLAS_ORDER order, CBLAS_TRANSPOSE TransA, CBLAS_TRANSPOSE TransB, const unsigned int M, const unsigned int N, const unsigned int K, const float alpha, const float *A, const unsigned int lda, const float *B, const unsigned int ldb, const float beta, float *C, const unsigned int ldc)
- static unsigned int **isamax_raw** (const unsigned int N, const float *X, const int incX)
- void **saxpy** (const unsigned int N, const float alpha, const float *X, const int incX, float *Y, const int incY)
- void **sgemm** (CBLAS_ORDER order, CBLAS_TRANSPOSE TransA, CBLAS_TRANSPOSE TransB, const unsigned int M, const unsigned int N, const unsigned int K, const float alpha, const float *A, const unsigned int lda, const float *B, const unsigned int ldb, const float beta, float *C, const unsigned int ldc)
- void **scopy** (const unsigned int N, const float *X, const int incX, float *Y, const int incY)
- void **sscal** (const int N, const float alpha, float *X, const int incX)
- float **snrm2** (const int N, const float *X, const int incX)
- float **sdot** (const unsigned int N, const float *X, const unsigned int incX, const float *Y, const unsigned int incY)
- void **sgemv** (CBLAS_ORDER order, CBLAS_TRANSPOSE TransA, const unsigned int M, const unsigned int N, const float alpha, const float *A, const unsigned int lda, const float *X, const int incX, const float beta, float *Y, const int incY)
- unsigned int **isamax** (const unsigned int N, const float *X, const int incX)
- static Tensor * **requestTensor_** (const TensorSpecV2 &spec, const GraphNode::ExecutionOrder &exec_order, const std::string &scope, TensorPool &tp, bool expose, bool trainable)
- template<typename T >
  static bool overlap (T s1, T e1, T s2, T e2)

  check if the two given intervals overlap
- static void validateIntervalOverlap (const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, const std::vector< size_t > &memory_size, const std::vector< size_t > &memory_offset, size_t memory_req)

  check if validate interval is overlapping in a very naive way.
- static void validateIntervalOverlap (const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, const std::vector< size_t > &memory_size, const std::vector< size_t > &memory_offset, size_t memory_req)

  check if validate interval is overlapping in a very naive way.
- static size_t computeSpace (unsigned int exec_order, std::vector< MemoryRequest * > &sorted_req, std::vector< std::pair< size_t, size_t >> &vacant)
- static void validateIntervalOverlap (const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, const std::vector< size_t > &memory_size, const std::vector< size_t > &memory_offset, size_t memory_req)

  check if validate interval is overlapping in a very naive way.
- std::ostream & **operator<<** (std::ostream &out, Tensor const &m)
- template<> std::unique_ptr< std::vector< std::pair< std::string, std::string > > > **Exporter::getResult**< **ml::train::ExportMethods::METHOD_STRINGVECTOR** > ()
- template<size_t I = 0, typename Callable , typename... Ts>
  std::enable_if< I==sizeof...(Ts), void >::type iterate_prop (Callable &&c, const std::tuple< Ts... > &tup)

  base case of iterate_prop, iterate_prop iterates the given tuple
- template<size_t I = 0, typename Callable , typename... Ts>
  std::enable_if<(I< sizeof...(Ts)), void >::type iterate_prop (Callable &&c, const std::tuple< Ts... > &tup)

  base case of iterate_prop, iterate_prop iterates the given tuple
- template<size_t I = 0, typename Callable , typename... Ts>
  std::enable_if<(I< sizeof...(Ts)), void >::type iterate_prop (Callable &&c, std::tuple< Ts... > &tup)

  <size_t I = 0, typename Callable, typename... Ts>
- template<typename Tuple >
  std::vector< std::string > loadProperties (const std::vector< std::string > &string_vector, Tuple &&props)

  load property from the api formatted string ({"key=value", "key1=value1"})
- static std::uniform_real_distribution< float > **dist** (-0.5, 0.5)

- unsigned int **getSeed** ()
- float **sqrtFloat** (float x)
- double **sqrtDouble** (double x)
- float **logFloat** (float x)
- float **exp_util** (float x)
- Tensor **rotate_180** (Tensor in)
- bool **isFileExist** (std::string file_name)
- template<typename T >
  static void **checkFile** (const T &file, const char *error_msg)
- void **checkedRead** (std::ifstream &file, char *array, std::streamsize size, const char *error_msg)
- void **checkedWrite** (std::ostream &file, const char *array, std::streamsize size, const char *error_msg)
- std::string **readString** (std::ifstream &file, const char *error_msg)
- void **writeString** (std::ofstream &file, const std::string &str, const char *error_msg)
- bool **endswith** (const std::string &target, const std::string &suffix)
- int **getKeyValue** (const std::string &input_str, std::string &key, std::string &value)
- int **getValues** (int n_str, std::string str, int *value)
- std::vector< std::string > **split** (const std::string &s, const std::regex &reg)
- bool **istrequal** (const std::string &a, const std::string &b)
- char * **getRealpath** (const char *name, char *resolved)
- tm * **getLocaltime** (tm *tp)

Variables

- std::mutex **factory_mutex**
- std::once_flag **global_app_context_init_flag**
- template<size_t I = 0, typename V , typename... Ts>
  std::enable_if< I< sizeof...(Ts), void >::type inline parse_properties(V &props, std::tuple< Ts... > &tup) { std::string name=std::get< I >tup);std::string prop=getConfig(name);if(!prop.empty()) props.↩ push_back(name+"="+prop);parse_properties< I+1 >props, tup);}const std::vector< std::string > AppContext::getProperties(void) { std::vector< std::string > properties;auto props=std::tuple("memory_↩ swap", "memory_swap_path");parse_properties(properties, props);return properties;}int AppContext::registerLayer(const std::string &library_path, const std::string &base_path) { const std::string full_path=getFullPath(library_↩ path, base_path);void *handle=dlopen(full_path.c_str(), RTLD_LAZY|RTLD_LOCAL);const char *error↩ _msg=dlerror();<< func_tag<< "open plugin failed, reason: "<< error_msg;nntrainer::Layer↩ Pluggable *pluggable=reinterpret_cast< nntrainer::LayerPluggable * > dlsym(handle, "ml_train_layer↩ _pluggable"));error_msg=dlerror();auto close_dl=[handle] { dlclose(handle);};NNTR_THROW_IF_CLEA↩ NUP(error_msg !=nullptr||pluggable==nullptr, std::invalid_argument, close_dl)<< func_tag<< "loading symbol failed, reason: "<< error_msg;auto layer=pluggable-> createfunc ()

  base case of iterate_prop, iterate_prop iterates the given tuple
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_IN_IDX** = 0
- constexpr char **USER_DATA** [ ] = "user_data"
- constexpr static float **NEGATIVE_SLOPE** = 0.01f
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0

- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr int **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static constexpr size_t **SINGLE_INOUT_IDX** = 0
- static auto **rng**
- constexpr const unsigned int **CONV2D_DIM** = 2
- constexpr const unsigned int **POOLING2D_DIM** = 2
- static auto **rng**

7.1.1 Detailed Description

PROFILE

7.1.2 Typedef Documentation

7.1.2.1 VarGradSpec

```
typedef std::tuple<TensorDim, Tensor::Initializer, bool, const std::string, TensorLifespan>
nntrainer::VarGradSpec
```

Specification of the Var_Grad (trainable tensor) as a tensor wrapper.

The tuple values are dimension, initializer, need_gradient property, the name, and lifespan of the Var_Grad object.

Definition at line 84 of file tensor_wrap_specs.h.

7.1.2.2 WeightSpec

```
typedef std::tuple<TensorDim, Tensor::Initializer, WeightRegularizer, float, float, float,
bool, const std::string> nntrainer::WeightSpec
```

Specification of the Weight as a tensor wrapper.

The tuple values are dimension, initializer, regularizer, regularizer_constant, decay, clip gradient constant, need_↩ gradient property amd name of the tensor object.

Definition at line 74 of file tensor_wrap_specs.h.

7.1.3 Enumeration Type Documentation

7.1.3.1 ActivationType

```
enum nntrainer::ActivationType  [strong]
```

Enumeration of activation function type.

Note

Upon changing this enum, ActivationTypeInfo must be changed accordingly

Enumerator

ACT_TANH	tanh
ACT_SIGMOID	sigmoid
ACT_RELU	ReLU
ACT_SWISH	Swish
ACT_SOFTMAX	softmax
ACT_LEAKY_RELU	Leaky ReLU
ACT_NONE	no op
ACT_UNKNOWN	unknown

Definition at line 32 of file common_properties.h.

7.1.3.2 AttentionParams [1/2]

```
enum nntrainer::AttentionParams
```

Enumerator

value	**Todo** make this property
attention_weight	intended comment for later use of attention_mask

Definition at line 29 of file attention_layer.cpp.

7.1.3.3 AttentionParams [2/2]

`enum nntrainer::AttentionParams`

Enumerator

value	**Todo** make this property
attention_weight	intended comment for later use of attention_mask

Definition at line 48 of file multi_head_attention_layer.cpp.

7.1.3.4 CachePolicy

`enum nntrainer::CachePolicy`

Enumerator

WRITE_BACK	invalidate will write to device
NO_WRITE_BACK	invalidate will not write to device
READ_CONSIST	validate will read from device
NO_READ_CONSIST	validate will not read from device
ALWAYS_SYNCED	Always synchronized with device
TEMPORAL	Will not be synchronized with device
FIRST_LAST_SKIP	Will skip first read and last write
ITERATION_CONSIST	Will skip first read and last write. other behaviors will be same as ALWAYS_SYNCED

Definition at line 25 of file cache_elem.h.

7.1.3.5 ExecutionMode

`enum nntrainer::ExecutionMode [strong]`

class telling the execution mode of the model/operation

Enumerator

INFERENCE	Training mode, label is necessary
VALIDATE	Inference mode, label is optional Validate mode, label is necessary

Definition at line 22 of file execution_mode.h.

7.1.3.6 INOUT_INDEX

`enum nntrainer::INOUT_INDEX`

Enumerator

QUERY	input index
OUTPUT	output index

Definition at line 37 of file multi_head_attention_layer.cpp.

7.1.3.7 InPlace

`enum nntrainer::InPlace [strong]`

Enum class for the various types of inplace modes supported by layer.

Enumerator

NONE	layer is not inplace
RESTRICTING	layer is in-place and does place restriction on layers ahead of it to be in-place
NON_RESTRICTING	layer is in-place and does NOT place restriction on the layers ahead of it to be in-place

Definition at line 59 of file layer_node.h.

7.1.3.8 MoLAttentionParams

`enum nntrainer::MoLAttentionParams`

Enumerator

value	**Todo** make this property

Definition at line 36 of file mol_attention_layer.cpp.

7.1.3.9 PrintOption

enum nntrainer::PrintOption

Print Options when printing layer info.

Enumerator

PRINT_INST_INFO	Option to print type & instance address info
PRINT_SHAPE_INFO	Option to print shape information, invalid before initiation
PRINT_PROP	Option to print properties
PRINT_PROP_META	Option to print properties that describe meta info e.g) layer activation type for non-activation layer.
PRINT_WEIGHTS	Option to print weights
PRINT_METRIC	Option to print metrics (currently loss only)

Definition at line 700 of file layer_node.cpp.

7.1.3.10 TensorLifespan

enum nntrainer::TensorLifespan [strong]

define the lifespan of the given tensor to reduce peak memory

Enumerator

UNMANAGED	tensor with no lifespan, will not be allocated
FORWARD_FUNC_LIFESPAN	tensor must not be reset before during the forward function call, eg. temporary tensors needed during forward operations
CALC_DERIV_LIFESPAN	must be valid during calcDerivative()
CALC_GRAD_LIFESPAN	tensor must be valid during calcGradient()
CALC_AGRAD_LIFESPAN	tensor must be valid during calcGradient()
CALC_GRAD_DERIV_LIFESPAN	tensor must not be reset before during the calc_grad and clac_deriv call, eg. temporary tensors needed during backward operations
CALC_GRAD_DERIV_AGRAD_LIFESPAN	tensor must not be reset before during the calc_grad, clac_deriv and apply gradient call, eg. temporary tensors needed during backward operations
FORWARD_GRAD_LIFESPAN	Forward + grad lifespan
FORWARD_GRAD_AGRAD_LIFESPAN	Forward + grad + apply gradient lifespan
FORWARD_DERIV_LIFESPAN	Forward + deriv lifespan
BACKWARD_FUNC_LIFESPAN	Alias of CALC_GRAD_DERIV_AGRAD_LIFESPAN
ITERATION_LIFESPAN	tensor must not be reset until the owning layer finishes its execution in the current iteration, eg. hidden memory/cells of RNN

Enumerator

EPOCH_LIFESPAN	tensor must be valid before the epoch ends
MAX_LIFESPAN	tensor must not be reset until the end of the model execution, eg. layer weights

Definition at line 38 of file tensor_wrap_specs.h.

7.1.3.11 WeightRegularizer

```
enum nntrainer::WeightRegularizer  [strong]
```

Enumeration of Weight Regularizer.

Todo Update to TensorRegularizer

Enumerator

L2NORM	L2 norm regularization
NONE	no regularization
UNKNOWN	Unknown

Definition at line 28 of file tensor_wrap_specs.h.

7.1.4 Function Documentation

7.1.4.1 add_default_object()

```
static void nntrainer::add_default_object (
            AppContext & ac )  [static]
```

Note

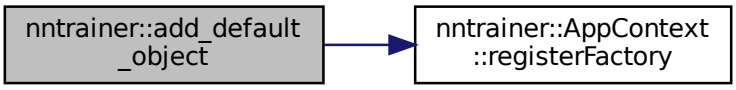
all layers should be added to the app_context to gaurantee that createLayer/createOptimizer class is created

proprocess layers

register losses

Definition at line 223 of file app_context.cpp.

Here is the call graph for this function:



7.1.4.2 AppContext::registerFactory< ml::train::LearningRateScheduler >()

```
const template int nntrainer::AppContext::registerFactory< ml::train::LearningRateScheduler >
(
            const FactoryType< ml::train::LearningRateScheduler > factory,
            const std::string & key,
            const int int_key )
```

int AppContext::registerFactory

int AppContext::registerFactory

7.1.4.3 AppContext::registerFactory< nntrainer::Layer >()

```
const template int nntrainer::AppContext::registerFactory< nntrainer::Layer > (
            const FactoryType< nntrainer::Layer > factory,
            const std::string & key,
            const int int_key )
```

int AppContext::registerFactory

int AppContext::registerFactory

7.1.4.4 AppContext::registerFactory< nntrainer::Optimizer >()

```
const template int nntrainer::AppContext::registerFactory< nntrainer::Optimizer > (
            const FactoryType< nntrainer::Optimizer > factory,
            const std::string & key,
            const int int_key )
```

int AppContext::registerFactory

int AppContext::registerFactory

7.1.4.5 buildTrasposeString()

```
static std::string nntrainer::buildTrasposeString (
            const std::array< props::PermuteDims, 3 > & arr )  [static]
```

buildTransposeString based on array

Todo deprecate this

Parameters

arr	array to make a representation

Returns

const std::string string to return

Definition at line 40 of file permute_layer.cpp.

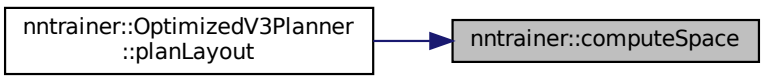
7.1.4.6 computeSpace()

```
static size_t nntrainer::computeSpace (
            unsigned int exec_order,
            std::vector< MemoryRequest * > & sorted_req,
            std::vector< std::pair< size_t, size_t >> & vacant )  [static]
```

TODO: try this

Definition at line 48 of file optimized_v3_planner.cpp.

Here is the caller graph for this function:



7.1.4.7 createLayerNode() [1/3]

```
std::unique_ptr< LayerNode > nntrainer::createLayerNode (
            const ml::train::LayerType & type,
            const std::vector< std::string > & properties = {} )
```

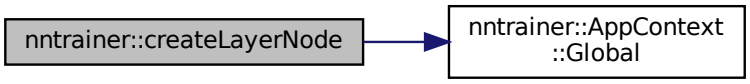
Layer factory creator with constructor.

LayerNode creator with constructor.

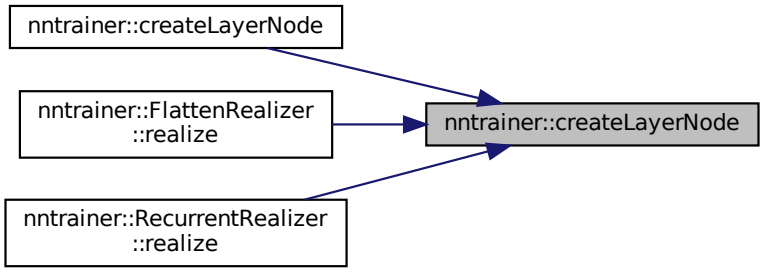
@params[in] type Type of the layer to be constructed @params[in] properties Properties of the layer

Definition at line 131 of file layer_node.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.4.8 createLayerNode() [2/3]

```
std::unique_ptr< LayerNode > nntrainer::createLayerNode (
            const std::string & type,
            const std::vector< std::string > & properties = {} )
```

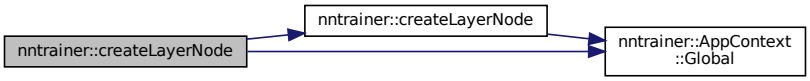
Layer factory creator with constructor.

LayerNode creator with constructor.

@params[in] type Type of the layer to be constructed @params[in] properties Properties of the layer

Definition at line 141 of file layer_node.cpp.

Here is the call graph for this function:



7.1.4.9 createLayerNode() [3/3]

```
std::unique_ptr< LayerNode > nntrainer::createLayerNode (
            std::unique_ptr< nntrainer::Layer > && layer,
            const std::vector< std::string > & properties )
```

Layer factory creator with constructor.

LayerNode creator with constructor.

@params[in] layer Already constructed layer @params[in] properties Properties of the layer

Definition at line 151 of file layer_node.cpp.

7.1.4.10 fini_global_context_nntrainer()

```
static void nntrainer::fini_global_context_nntrainer (
            void  )  [static]
```

finialize global context

Definition at line 219 of file app_context.cpp.

7.1.4.11 grucell_calcGradient()

```
static void nntrainer::grucell_calcGradient (
            const unsigned int unit,
            const unsigned int batch_size,
            const bool disable_bias,
            const bool integrate_bias,
            const bool reset_after,
            ActiFunc & acti_func,
            ActiFunc & recurrent_acti_func,
            const Tensor & input,
            const Tensor & prev_hidden_state,
            Tensor & d_prev_hidden_state,
            const Tensor & d_hidden_state,
            Tensor & d_weight_ih,
            const Tensor & weight_hh,
            Tensor & d_weight_hh,
            Tensor & d_bias_h,
            Tensor & d_bias_ih,
            const Tensor & bias_hh,
            Tensor & d_bias_hh,
            const Tensor & zrg,
            Tensor & d_zrg )  [static]
```

gru calcGradient

Definition at line 133 of file grucell.cpp.

7.1.4.12 grucell_forwarding()

```
static void nntrainer::grucell_forwarding (
             const unsigned int unit,
             const unsigned int batch_size,
             const bool disable_bias,
             const bool integrate_bias,
             const bool reset_after,
             ActiFunc & acti_func,
             ActiFunc & recurrent_acti_func,
             const Tensor & input,
             const Tensor & prev_hidden_state,
             Tensor & hidden_state,
             const Tensor & weight_ih,
             const Tensor & weight_hh,
             const Tensor & bias_h,
             const Tensor & bias_ih,
             const Tensor & bias_hh,
             Tensor & zrg )  [static]
```

gru forwarding

Definition at line 46 of file grucell.cpp.

7.1.4.13 iterate_prop() [1/3]

```
template<size_t I = 0, typename Callable , typename...  Ts>
std::enable_if<I == sizeof...(Ts), void>::type nntrainer::iterate_prop (
             Callable && c,
             const std::tuple< Ts...  > & tup )
```

base case of iterate_prop, iterate_prop iterates the given tuple

Template Parameters

I	size of tuple(automated)
Callable	generic lambda to be called during iteration
Ts	types from tuple

Parameters

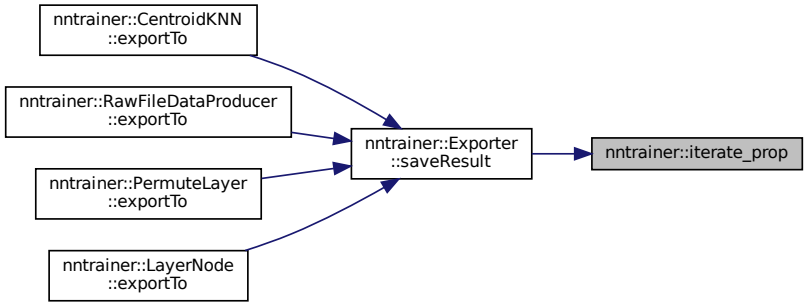
c	callable generic lambda
tup	tuple to be iterated

Returns

void

Definition at line 353 of file node_exporter.h.

Here is the caller graph for this function:



7.1.4.14 iterate_prop() [2/3]

```
template<size_t I = 0, typename Callable , typename...  Ts>
std::enable_if<(I < sizeof...(Ts)), void>::type nntrainer::iterate_prop (
            Callable && c,
            const std::tuple< Ts...  > & tup )
```

base case of iterate_prop, iterate_prop iterates the given tuple

Template Parameters

I	size of tuple(automated)
Callable	generic lambda to be called during iteration
Ts	types from tuple

Parameters

c	callable generic lambda
tup	tuple to be iterated

Returns

not used

Definition at line 369 of file node_exporter.h.

7.1.4.15 iterate_prop() [3/3]

```
template<size_t I = 0, typename Callable , typename...  Ts>
std::enable_if<(I < sizeof...(Ts)), void>::type nntrainer::iterate_prop (
```

```
            Callable && c,
            std::tuple< Ts...  > & tup )
```

<size_t I = 0, typename Callable, typename... Ts>

<size_t I = 0, typename Callable, typename... Ts> typename std::enable_if<(I < sizeof...(Ts)), void>::type iterate_prop(Callable &&c, const std::tuple<Ts...> &tup)

Definition at line 382 of file node_exporter.h.

7.1.4.16 loadProperties()

```
template<typename Tuple >
std::vector<std::string> nntrainer::loadProperties (
            const std::vector< std::string > & string_vector,
            Tuple && props )
```

load property from the api formatted string ({"key=value", "key1=value1"})

Template Parameters

Tuple	tuple type

Parameters

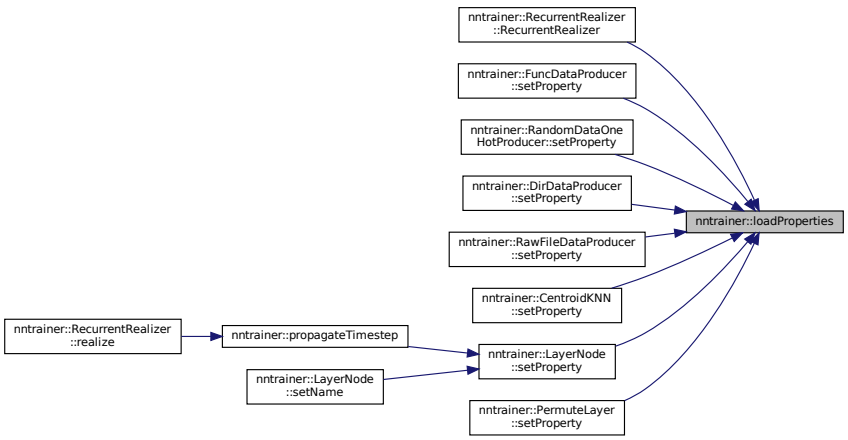
	string_vector	api formatted string;
out	*props*	props to be iterated

Returns

std::vector<std::string> vector of string that is not used while setting the property

Definition at line 400 of file node_exporter.h.

Here is the caller graph for this function:



7.1.4.17 overlap()

```
template<typename T >
static bool nntrainer::overlap (
            T s1,
            T e1,
            T s2,
            T e2 )  [static]
```

check if the two given intervals overlap

Parameters

s1	start of interval 1
e1	end of interval 1
s2	start of interval 2
e2	end of interval 2

Returns

true if overlap else false

Note

overlap check assumes are start is inclusive and end is exclusive

Definition at line 194 of file memory_pool.cpp.

7.1.4.18 parse_properties()

```
template<size_t I = 0, typename V , typename...  Ts>
std::enable_if<I == sizeof...(Ts), void>::type nntrainer::parse_properties (
            V & props,
            std::tuple< Ts...  > & tup )  [inline]
```

base case of iterate_prop, iterate_prop iterates the given tuple

Template Parameters

I	size of tuple(automated)
V	container type of properties
Ts	types from tuple

Parameters

prop	property container to be added to
tup	tuple to be iterated

Returns

void

Definition at line 424 of file app_context.cpp.

7.1.4.19 propagateTimestep()

```
static void nntrainer::propagateTimestep (
            LayerNode * node,
            unsigned int time_step,
            unsigned int max_time_step )  [static]
```

if node is of recurrent type, set time step and max time step

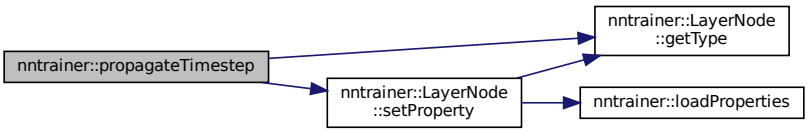
Parameters

node	node
time_step	time step
max_time_step	max time step

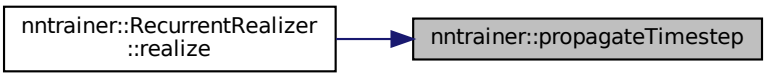
Todo add an interface to check if a layer supports a property

Definition at line 198 of file recurrent_realizer.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.4.20 setInplaceSharedMemoryConfigByLayer()

```
static void nntrainer::setInplaceSharedMemoryConfigByLayer (
            const std::shared_ptr< LayerNode > & lnode,
            bool & shared_var,
            bool & shared_grad )  [static]
```

Set the Inplace Shared Memory Config By Layer object.

Parameters

lnode	layer node object
shared_var	if the variable should be shared
shared_grad	if the gradient should be shared

for multiout layer, variables are shared but gradients are not

Todo for addition layer, variables are not shared but gradients are

Todo for layers which support in-place, both variables and gradients will be shared.

Todo add a check here is the layer being checked here can support in-place or not

Definition at line 684 of file network_graph.cpp.

7.1.4.21 suffixSpec()

```
static void nntrainer::suffixSpec (
            VarGradSpecV2 & spec,
            unsigned int idx )  [static]
```

rename specification

Parameters

spec	spec to rename
idx	idx to suffix

Definition at line 33 of file layer_context.cpp.

7.1.4.22 validateIntervalOverlap() [1/3]

```
static void nntrainer::validateIntervalOverlap (
            const std::vector< std::pair< unsigned int, unsigned int >> & memory_validity,
            const std::vector< size_t > & memory_size,
            const std::vector< size_t > & memory_offset,
            size_t memory_req )  [static]
```

check if validate interval is overlapping in a very naive way.

Parameters

memory_validity	validity
memory_size	size
memory_offset	offset
memory_req	request

Definition at line 56 of file optimized_v1_planner.cpp.

7.1.4.23 validateIntervalOverlap() [2/3]

```
static void nntrainer::validateIntervalOverlap (
            const std::vector< std::pair< unsigned int, unsigned int >> & memory_validity,
            const std::vector< size_t > & memory_size,
            const std::vector< size_t > & memory_offset,
            size_t memory_req )  [static]
```

check if validate interval is overlapping in a very naive way.

Parameters

memory_validity	validity
memory_size	size
memory_offset	offset
memory_req	request

Definition at line 69 of file optimized_v2_planner.cpp.

7.1.4.24 validateIntervalOverlap() [3/3]

```
static void nntrainer::validateIntervalOverlap (
            const std::vector< std::pair< unsigned int, unsigned int >> & memory_validity,
            const std::vector< size_t > & memory_size,
            const std::vector< size_t > & memory_offset,
            size_t memory_req )  [static]
```

check if validate interval is overlapping in a very naive way.

Parameters

memory_validity	validity
memory_size	size
memory_offset	offset
memory_req	request

Definition at line 87 of file optimized_v3_planner.cpp.

7.1.5 Variable Documentation

7.1.5.1 createfunc

```
template<size_t I = 0, typename V , typename...  Ts>
std::enable_if< I<sizeof...(Ts), void>::type inline parse_properties(V &props, std::tuple<Ts...>
&tup) { std::string name = std::get<I>tup); std::string prop = getConfig(name); if (!prop.↩
empty()) props.push_back(name + "=" + prop); parse_properties<I + 1>props, tup);}const std↩
::vector<std::string> AppContext::getProperties(void) { std::vector<std::string> properties;
auto props = std::tuple("memory_swap", "memory_swap_path"); parse_properties(properties, props);
return properties;}int AppContext::registerLayer(const std::string &library_path, const std↩
::string &base_path) { const std::string full_path = getFullPath(library_path, base_path);
void *handle = dlopen(full_path.c_str(), RTLD_LAZY | RTLD_LOCAL); const char *error_msg = dlerror();
<< func_tag << "open plugin failed, reason:  " << error_msg; nntrainer::LayerPluggable *pluggable
= reinterpret_cast<nntrainer::LayerPluggable *> dlsym(handle, "ml_train_layer_pluggable"));
error_msg = dlerror(); auto close_dl = [handle] { dlclose(handle); }; NNTR_THROW_IF_CLEAN↩
UP(error_msg != nullptr || pluggable == nullptr, std::invalid_argument, close_dl) << func_↩
tag << "loading symbol failed, reason:  " << error_msg; auto layer = pluggable-> nntrainer↩
::createfunc()
```

base case of iterate_prop, iterate_prop iterates the given tuple

Template Parameters

I	size of tuple(automated)
V	container type of properties
Ts	types from tuple

Parameters

prop	property container to be added to
tup	tuple to be iterated

Returns

void

Definition at line 480 of file app_context.cpp.

7.1.5.2 rng [1/2]

```
auto nntrainer::rng  [static]
```

Initial value:

```
= [] {
  std::mt19937 rng;
  rng.seed(getSeed());
  return rng;
}()
```

Definition at line 36 of file util_func.cpp.

7.1.5.3 rng [2/2]

```
auto nntrainer::rng  [static]
```

Initial value:

```
= [] {
  std::mt19937 rng;
  rng.seed(getSeed());
  return rng;
}()
```

Definition at line 89 of file tensor.cpp.

7.2 nntrainer::exception Namespace Reference

Classes

- class ErrorNotification

  Error Notification class, error is thrown when the class is destroyed. DO NOT use this outside as this contains throwing destructor.
- struct not_supported

  derived class of invalid argument to represent specific functionality not supported
- struct permission_denied

  derived class of invalid argument to represent permission is denied

7.2.1 Detailed Description

Note

underscore_case is used for ::exception to keep in accordance with std::exception

Chapter 8

Class Documentation

8.1 tflite::AbsOptionsBuilder Struct Reference

Public Types

- typedef AbsOptions **Table**

Public Member Functions

- **AbsOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- AbsOptionsBuilder & **operator=** (const AbsOptionsBuilder &)
- flatbuffers::Offset< AbsOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.1.1 Detailed Description

Definition at line 5948 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

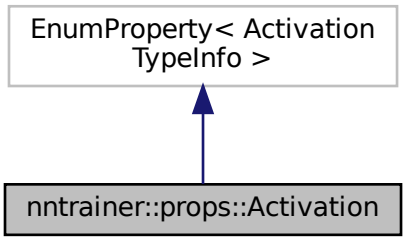
- compiler/tf_schema_generated.h

8.2 nntrainer::props::Activation Class Reference

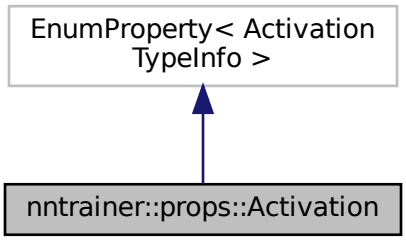
Activation Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Activation:



Collaboration diagram for nntrainer::props::Activation:



Public Types

- using **prop_tag** = enum_class_prop_tag

Static Public Attributes

- static constexpr const char * **key** = "activation"

8.2.1 Detailed Description

Activation Enumeration Information.

Definition at line 855 of file common_properties.h.

The documentation for this class was generated from the following file:

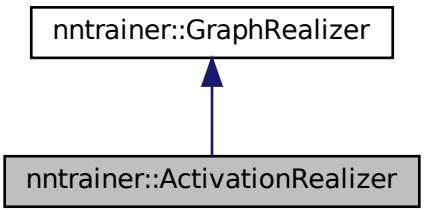
- layers/common_properties.h

8.3 nntrainer::ActivationRealizer Class Reference

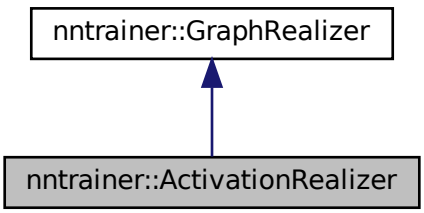
Graph realizer which realizes activation.

`#include <activation_realizer.h>`

Inheritance diagram for nntrainer::ActivationRealizer:



Collaboration diagram for nntrainer::ActivationRealizer:



Public Member Functions

- ~ActivationRealizer ()

  Destroy the Graph Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  graph realizer creates a new graph based on the reference

8.3.1 Detailed Description

Graph realizer which realizes activation.

Definition at line 27 of file activation_realizer.h.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 ~ActivationRealizer()

```
nntrainer::ActivationRealizer::~ActivationRealizer ( )
```

Destroy the Graph Realizer object.

Definition at line 24 of file activation_realizer.cpp.

8.3.3 Member Function Documentation

8.3.3.1 realize()

```
GraphRepresentation nntrainer::ActivationRealizer::realize (
            const GraphRepresentation & reference )  [override], [virtual]
```

graph realizer creates a new graph based on the reference

< layer_name

< act_layer_name

< temp_layer_name

< layer_name

realizing activation type not allowed because there is no good way to tell between 1. it is activation layer, 2. activation layer wants realization for now. later, we should have relu, sigmoid kind as layer type to resolve this matter, this is checked node->getActivationToBeRealized() but explicitly stated in order to make it robust, plus we might want to change the behavior

Implements nntrainer::GraphRealizer.

Definition at line 27 of file activation_realizer.cpp.

The documentation for this class was generated from the following files:

- compiler/activation_realizer.h
- compiler/activation_realizer.cpp

8.4 nntrainer::props::ActivationTypeInfo Struct Reference

Enumeration of activation function type.

`#include <common_properties.h>`

Public Types

- using **Enum** = nntrainer::ActivationType

Static Public Attributes

- static constexpr std::initializer_list< Enum > **EnumList**
- static constexpr const char * **EnumStr** [ ]

8.4.1 Detailed Description

Enumeration of activation function type.

Definition at line 840 of file common_properties.h.

8.4.2 Member Data Documentation

8.4.2.1 EnumList

```
constexpr std::initializer_list<Enum> nntrainer::props::ActivationTypeInfo::EnumList  [static],
[constexpr]
```

Initial value:

```
= {
    Enum::ACT_TANH,       Enum::ACT_SIGMOID, Enum::ACT_RELU, Enum::ACT_SOFTMAX,
    Enum::ACT_LEAKY_RELU, Enum::ACT_SWISH,   Enum::ACT_NONE, Enum::ACT_UNKNOWN}
```

Definition at line 842 of file common_properties.h.

8.4.2.2 EnumStr

```
constexpr const char* nntrainer::props::ActivationTypeInfo::EnumStr[]  [static], [constexpr]
```

Initial value:

```
= {"tanh",    "sigmoid",    "relu",
                                        "softmax", "leaky_relu", "swish",
                                        "none",    "unknown"}
```

Definition at line 846 of file common_properties.h.

The documentation for this struct was generated from the following file:

- layers/common_properties.h

8.5 tflite::AddNOptionsBuilder Struct Reference

Public Types

- typedef AddNOptions **Table**

Public Member Functions

- **AddNOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- AddNOptionsBuilder & **operator=** (const AddNOptionsBuilder &)
- flatbuffers::Offset< AddNOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.5.1 Detailed Description

Definition at line 6486 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.6 tflite::AddOptionsBuilder Struct Reference

Public Types

- typedef AddOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_pot_scale_int16** (bool pot_scale_int16)
- **AddOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- AddOptionsBuilder & **operator=** (const AddOptionsBuilder &)
- flatbuffers::Offset< AddOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.6.1 Detailed Description

Definition at line 3516 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.7 nntrainer::AppContext Class Reference

App.

Public Types

- using **PropsType** = std::vector< std::string >
- template<typename T >
  using **PtrType** = std::unique_ptr< T >
- template<typename T >
  using **FactoryType** = std::function< PtrType< T >(const PropsType &)>
- template<typename T >
  using **PtrFactoryType** = PtrType< T >(*)(const PropsType &)
- template<typename T >
  using **StrIndexType** = std::unordered_map< std::string, FactoryType< T > >
- using IntIndexType = std::unordered_map< int, std::string >
- template<typename T >
  using IndexType = std::tuple< StrIndexType< T >, IntIndexType >
- template<typename... Ts>
  using **FactoryMap** = std::tuple< IndexType< Ts >... >

Public Member Functions

- AppContext ()=default

  Default constructor.
- void setWorkingDirectory (const std::string &base)

  Set Working Directory for a relative path. working directory is set canonically.
- void unsetWorkingDirectory ()

  unset working directory
- bool hasWorkingDirectory ()

  query if the appcontext has working directory set
- int registerLayer (const std::string &library_path, const std::string &base_path="")

  *register a layer factory from a shared library plugin must have **extern "C" LayerPluggable *ml_train_layer_pluggable** defined else error*
- int registerOptimizer (const std::string &library_path, const std::string &base_path="")

  *register a optimizer factory from a shared library plugin must have **extern "C" OptimizerPluggable *ml_train_optimizer_pluggable** defined else error*
- std::vector< int > registerPluggableFromDirectory (const std::string &base_path)

  register pluggables from a directory.
- const std::string getWorkingPath (const std::string &path="")

  Get Working Path from a relative or representation of a path starting from working_path_base.

- const std::vector< std::string > getProperties (void)

  Get memory swap file path from configuration file.
- template<typename T >
  const int registerFactory (const PtrFactoryType< T > factory, const std::string &key="", const int int_key=-1)

  Factory register function, use this function to register custom object.
- template<typename T >
  const int registerFactory (const FactoryType< T > factory, const std::string &key="", const int int_key=-1)

  Factory register function, use this function to register custom object.
- template<typename T >
  PtrType< T > createObject (const int int_key, const PropsType &props={}) const

  Create an Object from the integer key.
- template<typename T >
  PtrType< T > createObject (const std::string &key, const PropsType &props={}) const

  Create an Object from the string key.

Static Public Member Functions

- static AppContext & Global ()

  Get Global app context.
- template<typename T >
  static PtrType< T > unknownFactory (const PropsType &props)

  special factory that throws for unknown

8.7.1 Detailed Description

App.

configuration

Definition at line 45 of file app_context.h.

8.7.2 Member Typedef Documentation

8.7.2.1 IndexType

```
template<typename T >
using nntrainer::AppContext::IndexType = std::tuple<StrIndexType<T>, IntIndexType>
```

This type contains tuple of 1) integer -> string index 2) string -> factory index

Definition at line 68 of file app_context.h.

8.7.2.2 IntIndexType

```
using nntrainer::AppContext::IntIndexType = std::unordered_map<int, std::string>
```

integer to string key

Definition at line 60 of file app_context.h.

8.7.3 Member Function Documentation

8.7.3.1 createObject() [1/2]

```
template<typename T >
PtrType<T> nntrainer::AppContext::createObject (
             const int int_key,
             const PropsType & props = {} ) const  [inline]
```

Create an Object from the integer key.

Template Parameters

T	Type of Object, currently, Only optimizer is supported

Parameters

int_key	integer key
props	property

Returns

PtrType<T> unique pointer to the object

Definition at line 213 of file app_context.h.

8.7.3.2 createObject() [2/2]

```
template<typename T >
PtrType<T> nntrainer::AppContext::createObject (
             const std::string & key,
             const PropsType & props = {} ) const  [inline]
```

Create an Object from the string key.

Template Parameters

T	Type of object, currently, only optimizer is supported

Parameters

key	integer key
props	property

Returns

PtrType<T> unique pointer to the object

Definition at line 240 of file app_context.h.

8.7.3.3 getProperties()

```
const std::vector<std::string> nntrainer::AppContext::getProperties (
            void  )
```

Get memory swap file path from configuration file.

Returns

memory swap path. If memory swap path is not presented in configuration file, it returns empty string

8.7.3.4 getWorkingPath()

```
const std::string nntrainer::AppContext::getWorkingPath (
            const std::string & path = "" )
```

Get Working Path from a relative or representation of a path starting from *working_path_base*.

Parameters

in	*path*	to make full path

Returns

If absolute path is given, returns *path* If relative path is given and working_path_base is not set, return relative path. If relative path is given and working_path_base has set, return absolute path from current working directory

Definition at line 409 of file app_context.cpp.

8.7.3.5 Global()

```
AppContext & nntrainer::AppContext::Global ( )  [static]
```

Get Global app context.

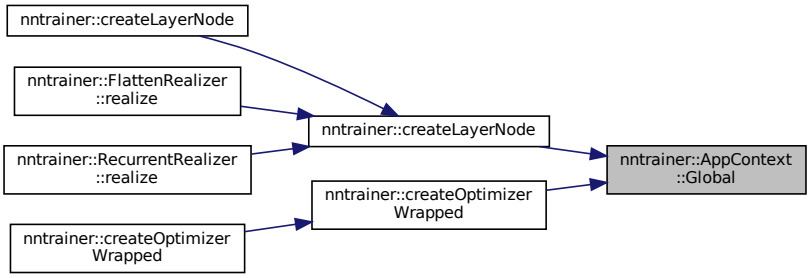
Returns

AppContext&

in g++ there is a bug that hangs up if caller throws, so registerer is noexcept although it'd better not https://gcc.gnu.org/bugzilla/show_bug.cgi?id=70298

Definition at line 377 of file app_context.cpp.

Here is the caller graph for this function:



8.7.3.6 hasWorkingDirectory()

```
bool nntrainer::AppContext::hasWorkingDirectory ( )  [inline]
```

query if the appcontext has working directory set

Return values

true	working path base is set
false	working path base is not set

Definition at line 105 of file app_context.h.

8.7.3.7 registerFactory() [1/2]

```
template<typename T >
const int nntrainer::AppContext::registerFactory (
```

```
            const FactoryType< T > factory,
            const std::string & key = "",
            const int int_key = -1 )
```

Factory register function, use this function to register custom object.

Template Parameters

T	object to create. Currently Optimizer, Layer is supported

Parameters

factory	factory function that creates std::unique_ptr<T>
key	key to access the factory, if key is empty, try to find key by calling factory({})->getType();
int_key	key to access the factory by integer, if it is -1(default), the function automatically unsigned the key and return

Returns

const int unique integer value to access the current factory

Exceptions

invalid	argument when key and/or int_key is already taken

8.7.3.8 registerFactory() [2/2]

```
template<typename T >
const int nntrainer::AppContext::registerFactory (
            const PtrFactoryType< T > factory,
            const std::string & key = "",
            const int int_key = -1 )  [inline]
```

Factory register function, use this function to register custom object.

Template Parameters

T	object to create. Currently Optimizer, Layer is supported

Parameters

factory	factory function that creates std::unique_ptr<T>
key	key to access the factory, if key is empty, try to find key by calling factory({})->getType();
int_key	key to access the factory by integer, if it is -1(default), the function automatically unsigned the key and return

Returns

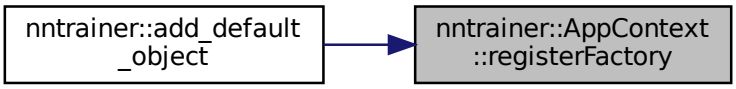
const int unique integer value to access the current factory

Exceptions

invalid	argument when key and/or int_key is already taken

Definition at line 179 of file app_context.h.

Here is the caller graph for this function:



8.7.3.9 registerLayer()

```
int nntrainer::AppContext::registerLayer (
             const std::string & library_path,
             const std::string & base_path = "" )
```

register a layer factory from a shared library plugin must have **extern "C" LayerPluggable ∗ml_train_layer_↩ pluggable** defined else error

Parameters

library_path	a file name of the library
base_path	base path to make a full path (optional)

Returns

int integer key to create the layer

Exceptions

std::invalid_parameter	if library_path is invalid or library is invalid

8.7.3.10 registerOptimizer()

```
int nntrainer::AppContext::registerOptimizer (
            const std::string & library_path,
            const std::string & base_path = "" )
```

register a optimizer factory from a shared library plugin must have **extern "C" OptimizerPluggable ∗ml_train_↩ optimizer_pluggable** defined else error

Parameters

library_path	a file name of the library
base_path	base path to make a full path (optional)

Returns

int integer key to create the optimizer

Exceptions

std::invalid_parameter	if library_path is invalid or library is invalid

8.7.3.11 registerPluggableFromDirectory()

```
std::vector<int> nntrainer::AppContext::registerPluggableFromDirectory (
            const std::string & base_path )
```

register pluggables from a directory.

Note

if you have a clashing type with already registered pluggable, it will throw from `registerFactory` function

Parameters

base_path	a directory path to search pluggables's

Returns

std::vector<int> list of integer key to create a pluggable

8.7.3.12 setWorkingDirectory()

```
void nntrainer::AppContext::setWorkingDirectory (
            const std::string & base )
```

Set Working Directory for a relative path. working directory is set canonically.

Parameters

in	*base*	base directory

Exceptions

std::invalid_argument	if path is not valid for current system

Definition at line 386 of file app_context.cpp.

8.7.3.13 unknownFactory()

```
template<typename T >
static PtrType<T> nntrainer::AppContext::unknownFactory (
             const PropsType & props )  [inline], [static]
```

special factory that throws for unknown

Template Parameters

T	object to create

Parameters

props	props to pass, not used

Exceptions

always	throw runtime_error

Definition at line 270 of file app_context.h.

8.7.3.14 unsetWorkingDirectory()

```
void nntrainer::AppContext::unsetWorkingDirectory ( )  [inline]
```

unset working directory

Definition at line 97 of file app_context.h.

The documentation for this class was generated from the following files:

- app_context.h
- app_context.cpp

8.8 tflite::ArgMaxOptionsBuilder Struct Reference

Public Types

- typedef ArgMaxOptions **Table**

Public Member Functions

- void **add_output_type** (tflite::TensorType output_type)
- **ArgMaxOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ArgMaxOptionsBuilder & **operator=** (const ArgMaxOptionsBuilder &)
- flatbuffers::Offset< ArgMaxOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.8.1 Detailed Description

Definition at line 5169 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.9 tflite::ArgMinOptionsBuilder Struct Reference

Public Types

- typedef ArgMinOptions **Table**

Public Member Functions

- void **add_output_type** (tflite::TensorType output_type)
- **ArgMinOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ArgMinOptionsBuilder & **operator=** (const ArgMinOptionsBuilder &)
- flatbuffers::Offset< ArgMinOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.9.1 Detailed Description

Definition at line 5211 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

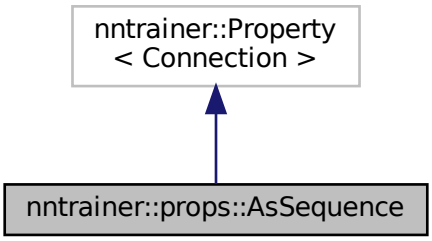
- compiler/tf_schema_generated.h

8.10 nntrainer::props::AsSequence Class Reference

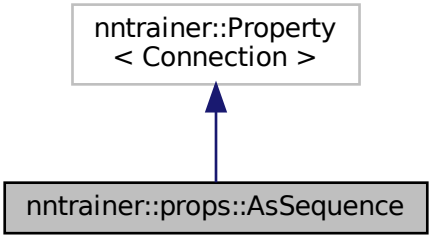
Identifiers to locate a connection which should be returned as whole used in recurrent realizer.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::AsSequence:



Collaboration diagram for nntrainer::props::AsSequence:



Public Types

- using **prop_tag** = connection_prop_tag

Static Public Attributes

- static constexpr const char * **key** = "as_sequence"

8.10.1 Detailed Description

Identifiers to locate a connection which should be returned as whole used in recurrent realizer.

Definition at line 664 of file common_properties.h.

The documentation for this class was generated from the following file:

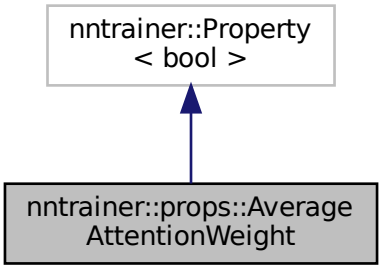
- layers/common_properties.h

8.11 nntrainer::props::AverageAttentionWeight Class Reference

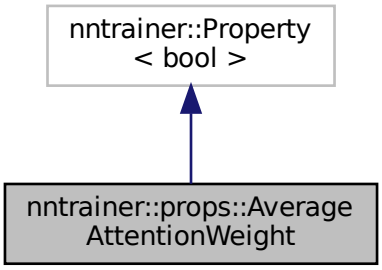
AverageAttentionWeight, average attention weight.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::AverageAttentionWeight:



Collaboration diagram for nntrainer::props::AverageAttentionWeight:



Public Types

- using prop_tag = bool_prop_tag

Static Public Attributes

- static constexpr const char * key

8.11.1 Detailed Description

AverageAttentionWeight, average attention weight.

Correspond with average_attn_weights of torch

Definition at line 1262 of file common_properties.h.

8.11.2 Member Typedef Documentation

8.11.2.1 prop_tag

```
using nntrainer::props::AverageAttentionWeight::prop_tag = bool_prop_tag
```

property type

Definition at line 1266 of file common_properties.h.

8.11.3 Member Data Documentation

8.11.3.1 key

```
constexpr const char* nntrainer::props::AverageAttentionWeight::key  [static], [constexpr]
```

Initial value:

```
=
    "average_attention_weight"
```

unique key to access

Definition at line 1264 of file common_properties.h.

The documentation for this class was generated from the following file:

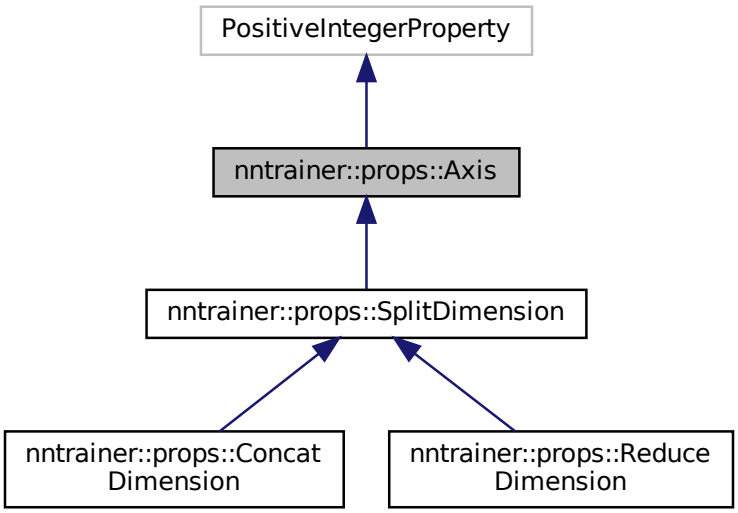
- layers/common_properties.h

8.12 nntrainer::props::Axis Class Reference

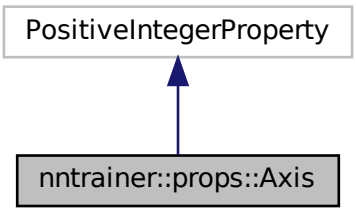
Axis property, idx in the dimension.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Axis:



Collaboration diagram for nntrainer::props::Axis:



Public Types

- using prop_tag = uint_prop_tag

Public Member Functions

- bool isValid (const unsigned int &value) const override
  check if given value is valid

Static Public Attributes

- static constexpr const char * key = "axis"

8.12.1 Detailed Description

Axis property, idx in the dimension.

Definition at line 269 of file common_properties.h.

8.12.2 Member Typedef Documentation

8.12.2.1 prop_tag

```
using nntrainer::props::Axis::prop_tag = uint_prop_tag
```

property type

Definition at line 272 of file common_properties.h.

8.12.3 Member Function Documentation

8.12.3.1 isValid()

```
bool nntrainer::props::Axis::isValid (
            const unsigned int & value ) const  [override]
```

check if given value is valid

Parameters

v	value to check

Return values

true	if it is greater equal to 0 and smaller than ml::train::TensorDim::MAXDIM

Return values

false	if it is samller than 0 or greater than ml::train::TensorDim::MAXDIM

Definition at line 99 of file common_properties.cpp.

8.12.4 Member Data Documentation

8.12.4.1 key

```
constexpr const char* nntrainer::props::Axis::key = "axis"  [static], [constexpr]
```

unique key to access

Definition at line 271 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.13 nntrainer::Backend Class Reference

Backend to be used for the operations to use.

```
#include <delegate.h>
```

Public Types

- enum BackendType { BackendType::armcl, BackendType::blas, BackendType::base }

  Backend type enumeration.

Public Member Functions

- Backend (BackendType backend=BackendType::base, int num_t=1)

  Construct a new Backend object.
- void setNumThreads (unsigned int num_threads)

  Set the number of threads for the backend if supported.

8.13.1 Detailed Description

Backend to be used for the operations to use.

Note

If some operation is not supported on set backend, that operation implemented with default backend will be used.

Definition at line 28 of file delegate.h.

8.13.2 Member Enumeration Documentation

8.13.2.1 BackendType

```
enum nntrainer::Backend::BackendType  [strong]
```

Backend type enumeration.

Enumerator

armcl	Arm Compute Library for backend
blas	Libopenblas for backend
base	Internally implemented operations for backend

Definition at line 33 of file delegate.h.

8.13.3 Constructor & Destructor Documentation

8.13.3.1 Backend()

```
nntrainer::Backend::Backend (
            BackendType backend = BackendType::base,
            int num_t = 1 )  [inline]
```

Construct a new Backend object.

Parameters

backend	Backend of be used. Defaults to base backend
num_t	Number of threads. Defaults to 1 thread

Definition at line 45 of file delegate.h.

8.13.4 Member Function Documentation

8.13.4.1 setNumThreads()

```
void nntrainer::Backend::setNumThreads (
            unsigned int num_threads )
```

Set the number of threads for the backend if supported.

Parameters

num_threads	Number of threads

The documentation for this class was generated from the following file:

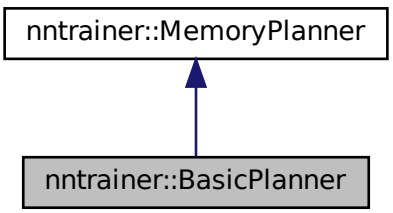
- delegate.h

8.14 nntrainer::BasicPlanner Class Reference

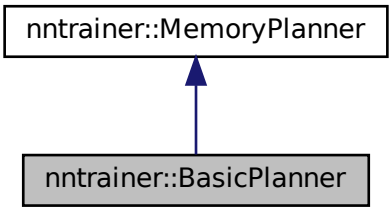
Basic Memory Planner provides the basic plan for memory layout.

```
#include <basic_planner.h>
```

Inheritance diagram for nntrainer::BasicPlanner:



Collaboration diagram for nntrainer::BasicPlanner:



Public Member Functions

- ~BasicPlanner ()=default

  BasicPlanner destructor.
- size_t planLayout (const std::vector< size_t > &memory_size, const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, std::vector< size_t > &memory_offset, std::vector< bool > &memory_is_wgrad, size_t n_wgrad=0) const
- const std::string & getType () const

  Get type of the planner.

Static Public Attributes

- static const std::string **type** = "basic_planner"

8.14.1 Detailed Description

Basic Memory Planner provides the basic plan for memory layout.

Basic planner performs no memory optimization and provides no memory sharing

Definition at line 29 of file basic_planner.h.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 ~BasicPlanner()

```
nntrainer::BasicPlanner::~BasicPlanner ( )  [default]
```

BasicPlanner destructor.

8.14.3 Member Function Documentation

8.14.3.1 getType()

```
const std::string& nntrainer::BasicPlanner::getType ( ) const  [inline], [virtual]
```

Get type of the planner.

Returns

The type of the planner

Implements nntrainer::MemoryPlanner.

Definition at line 55 of file basic_planner.h.

8.14.3.2 planLayout()

```
size_t nntrainer::BasicPlanner::planLayout (
            const std::vector< size_t > & memory_size,
            const std::vector< std::pair< unsigned int, unsigned int >> & memory_validity,
            std::vector< size_t > & memory_offset,
            std::vector< bool > & memory_is_wgrad,
            size_t n_wgrad = 0 ) const  [virtual]
```

The basic memory planner does not incorporate any memory sharing. This planner allocates independent memory for all the required memories without considering their memory validity.

Implements nntrainer::MemoryPlanner.

Definition at line 31 of file basic_planner.cpp.

The documentation for this class was generated from the following files:

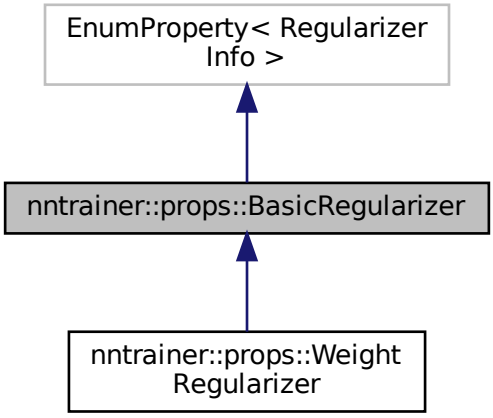
- tensor/basic_planner.h
- tensor/basic_planner.cpp

8.15 nntrainer::props::BasicRegularizer Class Reference

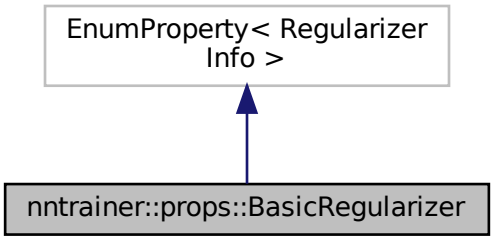
BasicRegularizer Regularization Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::BasicRegularizer:



Collaboration diagram for nntrainer::props::BasicRegularizer:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- BasicRegularizer (nntrainer::WeightRegularizer value)

  Construct a BasicRegularizer object.
- bool isValid (const nntrainer::WeightRegularizer &value) const override

  BasicRegularizer validator.

Static Public Attributes

- static constexpr const char * **key** = "basic_regularizer"

8.15.1 Detailed Description

BasicRegularizer Regularization Enumeration Information.

Definition at line 1011 of file common_properties.h.

8.15.2 Member Function Documentation

8.15.2.1 isValid()

```
bool nntrainer::props::BasicRegularizer::isValid (
            const nntrainer::WeightRegularizer & value ) const  [override]
```

BasicRegularizer validator.

Parameters

value	nntrainer::WeightRegularizer to validate

Return values

true	if value is not nntrainer::WeightRegularizer::UNKNOWN
false	if value is nntrainer::WeightRegularizer::UNKNOWN

Definition at line 320 of file common_properties.cpp.

The documentation for this class was generated from the following files:

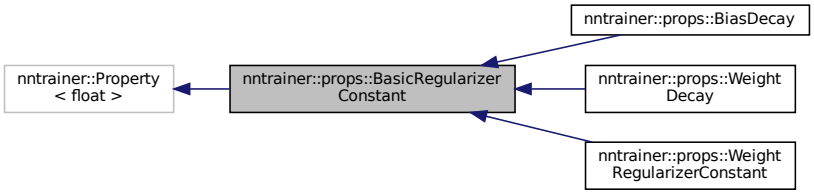
- layers/common_properties.h
- layers/common_properties.cpp

8.16 nntrainer::props::BasicRegularizerConstant Class Reference

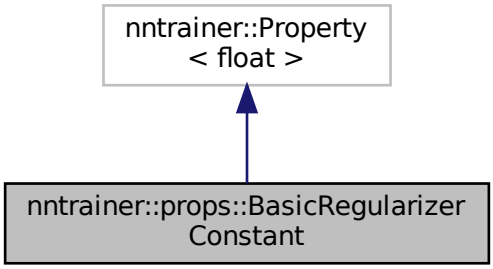
BasicRegularizerConstant property, this defines how much regularize the weight.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::BasicRegularizerConstant:



Collaboration diagram for nntrainer::props::BasicRegularizerConstant:



Public Types

- using prop_tag = float_prop_tag

Public Member Functions

- BasicRegularizerConstant (float value=1.0f)

  Construct a new BasicRegularizerConstant object.
- bool isValid (const float &value) const override

  check if given value is valid

Static Public Attributes

- static constexpr const char * key

8.16.1 Detailed Description

BasicRegularizerConstant property, this defines how much regularize the weight.

Definition at line 719 of file common_properties.h.

8.16.2 Member Typedef Documentation

8.16.2.1 prop_tag

```
using nntrainer::props::BasicRegularizerConstant::prop_tag = float_prop_tag
```

property type

Definition at line 729 of file common_properties.h.

8.16.3 Constructor & Destructor Documentation

8.16.3.1 BasicRegularizerConstant()

```
nntrainer::props::BasicRegularizerConstant::BasicRegularizerConstant (
            float value = 1.0f )
```

Construct a new BasicRegularizerConstant object.

Definition at line 270 of file common_properties.cpp.

8.16.4 Member Function Documentation

8.16.4.1 isValid()

```
bool nntrainer::props::BasicRegularizerConstant::isValid (
            const float & value ) const  [override]
```

check if given value is valid

Parameters

value	value to check

Returns

bool true if valid

Definition at line 279 of file common_properties.cpp.

8.16.5 Member Data Documentation

8.16.5.1 key

```
constexpr const char* nntrainer::props::BasicRegularizerConstant::key  [static], [constexpr]
```

Initial value:

```
=
    "basic_regularizer_constant"
```

unique key to access

Definition at line 727 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.17 tflite::BatchMatMulOptionsBuilder Struct Reference

Public Types

- typedef BatchMatMulOptions **Table**

Public Member Functions

- void **add_adj_x** (bool adj_x)
- void **add_adj_y** (bool adj_y)
- **BatchMatMulOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- BatchMatMulOptionsBuilder & **operator=** (const BatchMatMulOptionsBuilder &)
- flatbuffers::Offset< BatchMatMulOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.17.1 Detailed Description

Definition at line 7014 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.18 tflite::BatchToSpaceNDOptionsBuilder Struct Reference

Public Types

- typedef BatchToSpaceNDOptions **Table**

Public Member Functions

- **BatchToSpaceNDOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- BatchToSpaceNDOptionsBuilder & **operator=** (const BatchToSpaceNDOptionsBuilder &)
- flatbuffers::Offset< BatchToSpaceNDOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.18.1 Detailed Description

Definition at line 4256 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

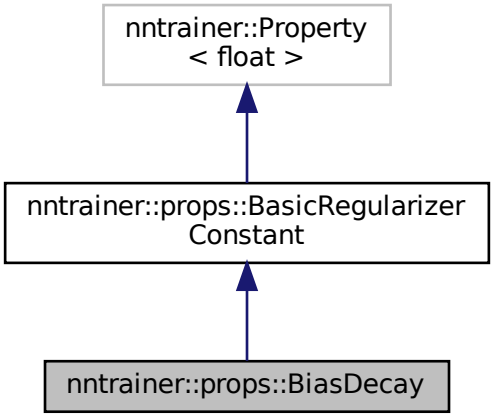
- compiler/tf_schema_generated.h

8.19 nntrainer::props::BiasDecay Class Reference

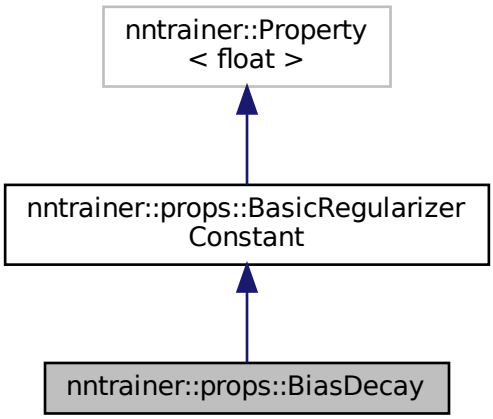
BiasDecay property, this defines how much regularize the weight.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::BiasDecay:



Collaboration diagram for nntrainer::props::BiasDecay:



Public Member Functions

- BiasDecay (float value=0.0f)

  Construct a new BiasDecay object.

Static Public Attributes

- static constexpr const char * key = "bias_decay"

Additional Inherited Members

8.19.1 Detailed Description

BiasDecay property, this defines how much regularize the weight.

Definition at line 779 of file common_properties.h.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 BiasDecay()

```
nntrainer::props::BiasDecay::BiasDecay (
            float value = 0.0f )
```

Construct a new BiasDecay object.

Definition at line 275 of file common_properties.cpp.

8.19.3 Member Data Documentation

8.19.3.1 key

```
constexpr const char* nntrainer::props::BiasDecay::key = "bias_decay"  [static], [constexpr]
```

unique key to access

Definition at line 787 of file common_properties.h.

The documentation for this class was generated from the following files:

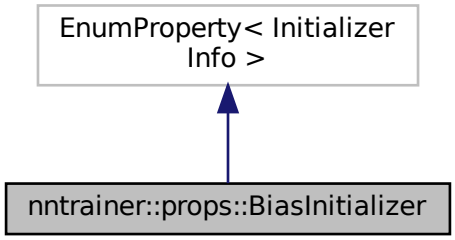
- layers/common_properties.h
- layers/common_properties.cpp

8.20 nntrainer::props::BiasInitializer Class Reference

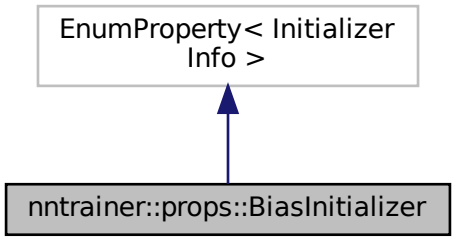
BiasInitializer Initialization Enumeration Information.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::BiasInitializer:



Collaboration diagram for nntrainer::props::BiasInitializer:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- BiasInitializer (Tensor::Initializer value=Tensor::Initializer::ZEROS)

  Construct a BiasInitializer object.

Static Public Attributes

- static constexpr const char * **key** = "bias_initializer"

8.20.1 Detailed Description

BiasInitializer Initialization Enumeration Information.

Definition at line 930 of file common_properties.h.

The documentation for this class was generated from the following files:

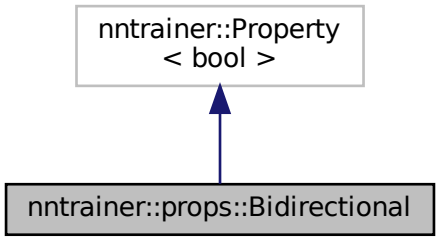
- layers/common_properties.h
- layers/common_properties.cpp

8.21 nntrainer::props::Bidirectional Class Reference

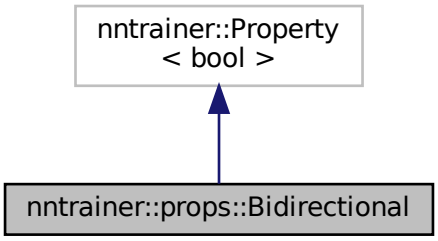
bidirectional property, used to make bidirectional layers

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Bidirectional:



Collaboration diagram for nntrainer::props::Bidirectional:



Public Types

- using **prop_tag** = bool_prop_tag

Public Member Functions

- Bidirectional (bool value=false)

  Construct a new Bidirectional object.

Static Public Attributes

- static constexpr const char * **key** = "bidirectional"

8.21.1 Detailed Description

bidirectional property, used to make bidirectional layers

Definition at line 648 of file common_properties.h.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 Bidirectional()

```
nntrainer::props::Bidirectional::Bidirectional (
            bool value = false )
```

Construct a new Bidirectional object.

Definition at line 81 of file common_properties.cpp.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.22 tflite::BidirectionalSequenceLSTMOptionsBuilder Struct Reference

Public Types

- typedef BidirectionalSequenceLSTMOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_cell_clip** (float cell_clip)
- void **add_proj_clip** (float proj_clip)
- void **add_merge_outputs** (bool merge_outputs)
- void **add_time_major** (bool time_major)
- void **add_asymmetric_quantize_inputs** (bool asymmetric_quantize_inputs)
- **BidirectionalSequenceLSTMOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- BidirectionalSequenceLSTMOptionsBuilder & **operator=** (const BidirectionalSequenceLSTMOptionsBuilder &)
- flatbuffers::Offset< BidirectionalSequenceLSTMOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.22.1 Detailed Description

Definition at line 3908 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.23 tflite::BidirectionalSequenceRNNOptionsBuilder Struct Reference

Public Types

- typedef BidirectionalSequenceRNNOptions **Table**

Public Member Functions

- void **add_time_major** (bool time_major)
- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_merge_outputs** (bool merge_outputs)
- void **add_asymmetric_quantize_inputs** (bool asymmetric_quantize_inputs)
- **BidirectionalSequenceRNNOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- BidirectionalSequenceRNNOptionsBuilder & **operator=** (const BidirectionalSequenceRNNOptionsBuilder &)
- flatbuffers::Offset< BidirectionalSequenceRNNOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.23.1 Detailed Description

Definition at line 3288 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

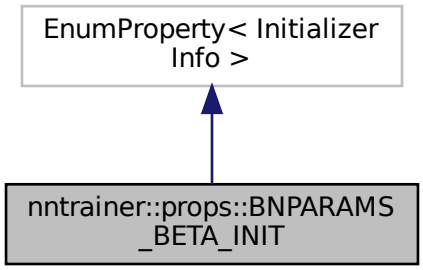
- compiler/tf_schema_generated.h

8.24 nntrainer::props::BNPARAMS_BETA_INIT Class Reference

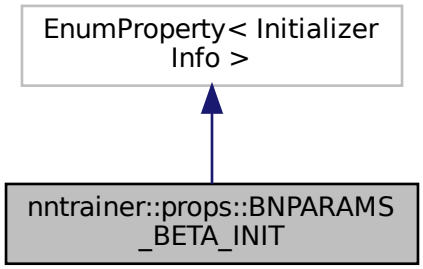
BNPARAMS_BETA_INIT Initialization Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::BNPARAMS_BETA_INIT:



Collaboration diagram for nntrainer::props::BNPARAMS_BETA_INIT:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- BNPARAMS_BETA_INIT (Tensor::Initializer value=Tensor::Initializer::ZEROS)

  Construct a BNPARAMS_BETA_INIT object.

Static Public Attributes

- static constexpr const char * **key** = "beta_initializer"

8.24.1 Detailed Description

BNPARAMS_BETA_INIT Initialization Enumeration Information.

Definition at line 986 of file common_properties.h.

The documentation for this class was generated from the following files:

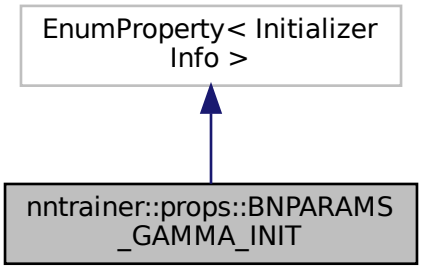
- layers/common_properties.h
- layers/common_properties.cpp

8.25 nntrainer::props::BNPARAMS_GAMMA_INIT Class Reference

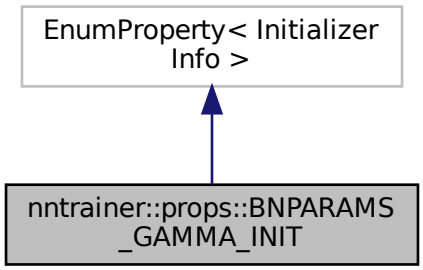
BNPARAMS_GAMMA_INIT Initialization Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::BNPARAMS_GAMMA_INIT:



Collaboration diagram for nntrainer::props::BNPARAMS_GAMMA_INIT:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- BNPARAMS_GAMMA_INIT (Tensor::Initializer value=Tensor::Initializer::ONES)

  Construct a BNPARAMS_GAMMA_INIT object.

Static Public Attributes

- static constexpr const char * **key** = "gamma_initializer"

8.25.1 Detailed Description

BNPARAMS_GAMMA_INIT Initialization Enumeration Information.

Definition at line 972 of file common_properties.h.

The documentation for this class was generated from the following files:

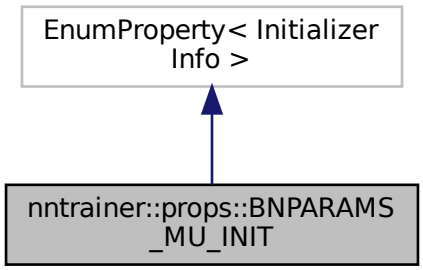
- layers/common_properties.h
- layers/common_properties.cpp

8.26 nntrainer::props::BNPARAMS_MU_INIT Class Reference

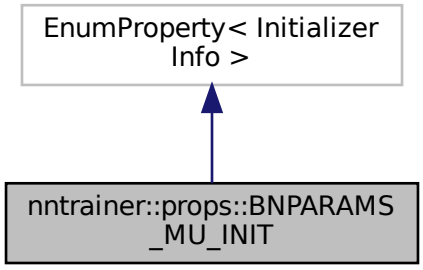
BNPARAMS_MU_INIT Initialization Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::BNPARAMS_MU_INIT:



Collaboration diagram for nntrainer::props::BNPARAMS_MU_INIT:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- BNPARAMS_MU_INIT (Tensor::Initializer value=Tensor::Initializer::ZEROS)

  Construct a BNPARAMS_MU_INIT object.

Static Public Attributes

- static constexpr const char * **key** = "moving_mean_initializer"

8.26.1 Detailed Description

BNPARAMS_MU_INIT Initialization Enumeration Information.

Definition at line 944 of file common_properties.h.

The documentation for this class was generated from the following files:

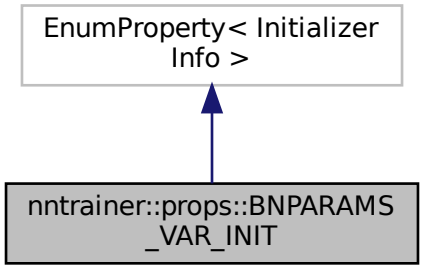
- layers/common_properties.h
- layers/common_properties.cpp

8.27 nntrainer::props::BNPARAMS_VAR_INIT Class Reference

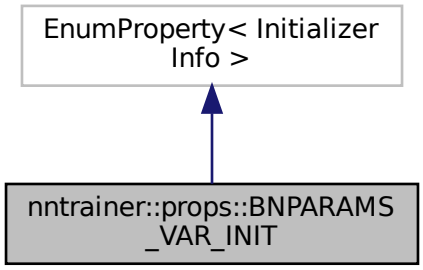
BNPARAMS_VAR_INIT Initialization Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::BNPARAMS_VAR_INIT:



Collaboration diagram for nntrainer::props::BNPARAMS_VAR_INIT:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- BNPARAMS_VAR_INIT (Tensor::Initializer value=Tensor::Initializer::ONES)

  Construct a BNPARAMS_VAR_INIT object.

Static Public Attributes

- static constexpr const char * **key** = "moving_variance_initializer"

8.27.1 Detailed Description

BNPARAMS_VAR_INIT Initialization Enumeration Information.

Definition at line 958 of file common_properties.h.

The documentation for this class was generated from the following files:

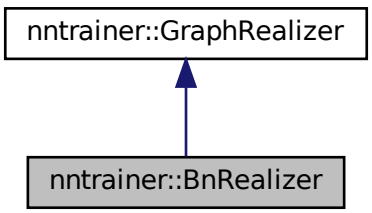
- layers/common_properties.h
- layers/common_properties.cpp

8.28 nntrainer::BnRealizer Class Reference

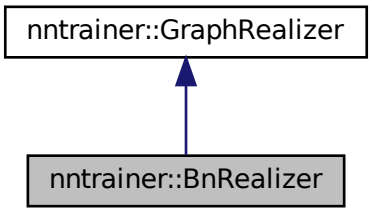
Graph realizer class which removes batch normalization layer from the graph.

`#include <bn_realizer.h>`

Inheritance diagram for nntrainer::BnRealizer:



Collaboration diagram for nntrainer::BnRealizer:



Public Member Functions

- BnRealizer ()=default

  Construct a new BN Realizer object.
- ~BnRealizer ()=default

  Destroy the Graph Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  graph realizer creates a shallow copied graph based on the reference

8.28.1 Detailed Description

Graph realizer class which removes batch normalization layer from the graph.

Note

This assumes the number of input / output connection of batch normalization layer == 1

Definition at line 32 of file bn_realizer.h.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 BnRealizer()

```
nntrainer::BnRealizer::BnRealizer ( )  [default]
```

Construct a new BN Realizer object.

8.28.2.2 ~BnRealizer()

```
nntrainer::BnRealizer::~BnRealizer ( )  [default]
```

Destroy the Graph Realizer object.

8.28.3 Member Function Documentation

8.28.3.1 realize()

```
GraphRepresentation nntrainer::BnRealizer::realize (
            const GraphRepresentation & reference )  [override], [virtual]
```

graph realizer creates a shallow copied graph based on the reference

Note

bn realizer removes batch normalization layers from GraphRepresentation

Parameters

reference	GraphRepresentation to be realized

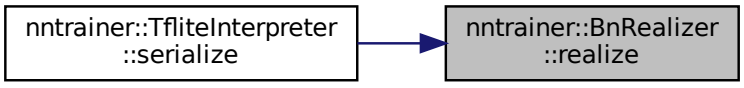
Exceptions

std::invalid_argument	if graph is ill formed

Implements nntrainer::GraphRealizer.

Definition at line 27 of file bn_realizer.cpp.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- compiler/bn_realizer.h
- compiler/bn_realizer.cpp

8.29 nntrainer::Tensor::BroadcastInfo Struct Reference

Public Member Functions

- BroadcastInfo ()

  Construct a new External Loop Info object.

Public Attributes

- unsigned int buffer_size
- int buffer_axis
- std::array< unsigned int, TensorDim::MAXDIM > strides

8.29.1 Detailed Description

Definition at line 74 of file tensor.cpp.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 BroadcastInfo()

```
nntrainer::Tensor::BroadcastInfo::BroadcastInfo ( )  [inline]
```

Construct a new External Loop Info object.

Definition at line 80 of file tensor.cpp.

8.29.3 Member Data Documentation

8.29.3.1 buffer_axis

```
int nntrainer::Tensor::BroadcastInfo::buffer_axis
```

the smallest axis that should be looped. -1 means no loop needed

Definition at line 83 of file tensor.cpp.

8.29.3.2 buffer_size

```
unsigned int nntrainer::Tensor::BroadcastInfo::buffer_size
```

virtual size of the buffer

Definition at line 82 of file tensor.cpp.

8.29.3.3 strides

```
std::array<unsigned int, TensorDim::MAXDIM> nntrainer::Tensor::BroadcastInfo::strides
```

modified strides for the loop

Definition at line 86 of file tensor.cpp.

The documentation for this struct was generated from the following file:

- tensor/tensor.cpp

8.30 tflite::BufferBuilder Struct Reference

Public Types

- typedef Buffer **Table**

Public Member Functions

- void **add_data** (flatbuffers::Offset< flatbuffers::Vector< uint8_t >> data)
- **BufferBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- BufferBuilder & **operator=** (const BufferBuilder &)
- flatbuffers::Offset< Buffer > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.30.1 Detailed Description

Definition at line 8184 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.31 tflite::BuiltinOptionsTraits< T > Struct Template Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_NONE

8.31.1 Detailed Description

template<typename T>
struct tflite::BuiltinOptionsTraits< T >

Definition at line 1295 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.32 tflite::BuiltinOptionsTraits< tflite::AbsOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_AbsOptions

8.32.1 Detailed Description

Definition at line 1607 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.33 tflite::BuiltinOptionsTraits< tflite::AddNOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_AddNOptions

8.33.1 Detailed Description

Definition at line 1623 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.34 tflite::BuiltinOptionsTraits< tflite::AddOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_AddOptions

8.34.1 Detailed Description

Definition at line 1339 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.35 tflite::BuiltinOptionsTraits< tflite::ArgMaxOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ArgMaxOptions

8.35.1 Detailed Description

Definition at line 1455 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.36 tflite::BuiltinOptionsTraits< tflite::ArgMinOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ArgMinOptions

8.36.1 Detailed Description

Definition at line 1523 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.37 tflite::BuiltinOptionsTraits< tflite::BatchMatMulOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_BatchMatMulOptions

8.37.1 Detailed Description

Definition at line 1699 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.38 tflite::BuiltinOptionsTraits< tflite::BatchToSpaceNDOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_BatchToSpaceNDOptions

8.38.1 Detailed Description

Definition at line 1391 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.39 tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceLSTMOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_BidirectionalSequenceLSTMOptions

8.39.1 Detailed Description

Definition at line 1571 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.40 tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceRNNOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_BidirectionalSequenceRNNOptions

8.40.1 Detailed Description

Definition at line 1575 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.41 tflite::BuiltinOptionsTraits< tflite::CallOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_CallOptions

8.41.1 Detailed Description

Definition at line 1359 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.42 tflite::BuiltinOptionsTraits< tflite::CastOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_CastOptions

8.42.1 Detailed Description

Definition at line 1443 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.43 tflite::BuiltinOptionsTraits< tflite::ConcatEmbeddingsOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ConcatEmbeddingsOptions

8.43.1 Detailed Description

Definition at line 1307 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.44 tflite::BuiltinOptionsTraits< tflite::ConcatenationOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ConcatenationOptions

8.44.1 Detailed Description

Definition at line 1335 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.45 tflite::BuiltinOptionsTraits< tflite::Conv2DOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_Conv2DOptions

8.45.1 Detailed Description

Definition at line 1299 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.46 tflite::BuiltinOptionsTraits< tflite::CosOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_CosOptions

8.46.1 Detailed Description

Definition at line 1631 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.47 tflite::BuiltinOptionsTraits< tflite::CumsumOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_CumsumOptions

8.47.1 Detailed Description

Definition at line 1703 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.48 tflite::BuiltinOptionsTraits< tflite::DensifyOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_DensifyOptions

8.48.1 Detailed Description

Definition at line 1691 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.49 tflite::BuiltinOptionsTraits< tflite::DepthToSpaceOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_DepthToSpaceOptions

8.49.1 Detailed Description

Definition at line 1671 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.50 tflite::BuiltinOptionsTraits< tflite::DepthwiseConv2DOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_DepthwiseConv2DOptions

8.50.1 Detailed Description

Definition at line 1303 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.51 tflite::BuiltinOptionsTraits< tflite::DequantizeOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_DequantizeOptions

8.51.1 Detailed Description

Definition at line 1447 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.52 tflite::BuiltinOptionsTraits< tflite::DivOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_DivOptions

8.52.1 Detailed Description

Definition at line 1411 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.53 tflite::BuiltinOptionsTraits< tflite::EmbeddingLookupSparseOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_EmbeddingLookupSparseOptions

8.53.1 Detailed Description

Definition at line 1375 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.54 tflite::BuiltinOptionsTraits< tflite::EqualOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_EqualOptions

8.54.1 Detailed Description

Definition at line 1507 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.55 tflite::BuiltinOptionsTraits< tflite::ExpandDimsOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ExpandDimsOptions

8.55.1 Detailed Description

Definition at line 1503 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.56 tflite::BuiltinOptionsTraits< tflite::ExpOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ExpOptions

8.56.1 Detailed Description

Definition at line 1427 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.57 tflite::BuiltinOptionsTraits< tflite::FakeQuantOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_FakeQuantOptions

8.57.1 Detailed Description

Definition at line 1527 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.58 tflite::BuiltinOptionsTraits< tflite::FillOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_FillOptions

8.58.1 Detailed Description

Definition at line 1567 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.59 tflite::BuiltinOptionsTraits< tflite::FloorDivOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_FloorDivOptions

8.59.1 Detailed Description

Definition at line 1555 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.60 tflite::BuiltinOptionsTraits< tflite::FloorModOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_FloorModOptions

8.60.1 Detailed Description

Definition at line 1583 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.61 tflite::BuiltinOptionsTraits< tflite::FullyConnectedOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_FullyConnectedOptions

8.61.1 Detailed Description

Definition at line 1327 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.62 tflite::BuiltinOptionsTraits< tflite::GatherNdOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_GatherNdOptions

8.62.1 Detailed Description

Definition at line 1627 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.63 tflite::BuiltinOptionsTraits< tflite::GatherOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_GatherOptions

8.63.1 Detailed Description

Definition at line 1387 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.64 tflite::BuiltinOptionsTraits< tflite::GreaterEqualOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_GreaterEqualOptions

8.64.1 Detailed Description

Definition at line 1475 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.65 tflite::BuiltinOptionsTraits< tflite::GreaterOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_GreaterOptions

8.65.1 Detailed Description

Definition at line 1471 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.66 tflite::BuiltinOptionsTraits< tflite::HardSwishOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_HardSwishOptions

8.66.1 Detailed Description

Definition at line 1659 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.67 tflite::BuiltinOptionsTraits< tflite::IfOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_IfOptions

8.67.1 Detailed Description

Definition at line 1663 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.68 tflite::BuiltinOptionsTraits< tflite::L2NormOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_L2NormOptions

8.68.1 Detailed Description

Definition at line 1343 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.69 tflite::BuiltinOptionsTraits< tflite::LeakyReluOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LeakyReluOptions

8.69.1 Detailed Description

Definition at line 1595 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.70 tflite::BuiltinOptionsTraits< tflite::LessEqualOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LessEqualOptions

8.70.1 Detailed Description

Definition at line 1479 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.71 tflite::BuiltinOptionsTraits< tflite::LessOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LessOptions

8.71.1 Detailed Description

Definition at line 1459 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.72 tflite::BuiltinOptionsTraits< tflite::LocalResponseNormalizationOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LocalResponseNormalizationOptions

8.72.1 Detailed Description

Definition at line 1347 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.73 tflite::BuiltinOptionsTraits< tflite::LogicalAndOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LogicalAndOptions

8.73.1 Detailed Description

Definition at line 1543 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.74 tflite::BuiltinOptionsTraits< tflite::LogicalNotOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LogicalNotOptions

8.74.1 Detailed Description

Definition at line 1547 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.75 tflite::BuiltinOptionsTraits< tflite::LogicalOrOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LogicalOrOptions

8.75.1 Detailed Description

Definition at line 1535 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.76 tflite::BuiltinOptionsTraits< tflite::LogSoftmaxOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LogSoftmaxOptions

8.76.1 Detailed Description

Definition at line 1439 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.77 tflite::BuiltinOptionsTraits< tflite::LSHProjectionOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LSHProjectionOptions

8.77.1 Detailed Description

Definition at line 1311 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.78 tflite::BuiltinOptionsTraits< tflite::LSTMOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_LSTMOptions

8.78.1 Detailed Description

Definition at line 1351 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.79 tflite::BuiltinOptionsTraits< tflite::MatrixDiagOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_MatrixDiagOptions

8.79.1 Detailed Description

Definition at line 1647 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.80 tflite::BuiltinOptionsTraits< tflite::MatrixSetDiagOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_MatrixSetDiagOptions

8.80.1 Detailed Description

Definition at line 1655 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.81 tflite::BuiltinOptionsTraits< tflite::MaximumMinimumOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_MaximumMinimumOptions

8.81.1 Detailed Description

Definition at line 1451 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.82 tflite::BuiltinOptionsTraits< tflite::MirrorPadOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_MirrorPadOptions

8.82.1 Detailed Description

Definition at line 1603 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.83 tflite::BuiltinOptionsTraits< tflite::MulOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_MulOptions

8.83.1 Detailed Description

Definition at line 1379 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.84 tflite::BuiltinOptionsTraits< tflite::NegOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_NegOptions

8.84.1 Detailed Description

Definition at line 1463 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.85 tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV4Options > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_NonMaxSuppressionV4Options

8.85.1 Detailed Description

Definition at line 1675 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.86 tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV5Options > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_NonMaxSuppressionV5Options

8.86.1 Detailed Description

Definition at line 1679 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.87 tflite::BuiltinOptionsTraits< tflite::NotEqualOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_NotEqualOptions

8.87.1 Detailed Description

Definition at line 1511 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.88 tflite::BuiltinOptionsTraits< tflite::OneHotOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_OneHotOptions

8.88.1 Detailed Description

Definition at line 1539 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.89 tflite::BuiltinOptionsTraits< tflite::PackOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_PackOptions

8.89.1 Detailed Description

Definition at line 1531 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.90 tflite::BuiltinOptionsTraits< tflite::PadOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_PadOptions

8.90.1 Detailed Description

Definition at line 1383 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.91 tflite::BuiltinOptionsTraits< tflite::PadV2Options > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_PadV2Options

8.91.1 Detailed Description

Definition at line 1467 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.92 tflite::BuiltinOptionsTraits< tflite::Pool2DOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_Pool2DOptions

8.92.1 Detailed Description

Definition at line 1315 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.93 tflite::BuiltinOptionsTraits< tflite::PowOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_PowOptions

8.93.1 Detailed Description

Definition at line 1519 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.94 tflite::BuiltinOptionsTraits< tflite::QuantizeOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_QuantizeOptions

8.94.1 Detailed Description

Definition at line 1651 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.95 tflite::BuiltinOptionsTraits< tflite::RangeOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_RangeOptions

8.95.1 Detailed Description

Definition at line 1587 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.96 tflite::BuiltinOptionsTraits< tflite::RankOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_RankOptions

8.96.1 Detailed Description

Definition at line 1639 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.97 tflite::BuiltinOptionsTraits< tflite::ReducerOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ReducerOptions

8.97.1 Detailed Description

Definition at line 1403 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.98 tflite::BuiltinOptionsTraits< tflite::ReshapeOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ReshapeOptions

8.98.1 Detailed Description

Definition at line 1363 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.99 tflite::BuiltinOptionsTraits< tflite::ResizeBilinearOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ResizeBilinearOptions

8.99.1 Detailed Description

Definition at line 1355 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.100 tflite::BuiltinOptionsTraits< tflite::ResizeNearestNeighborOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ResizeNearestNeighborOptions

8.100.1 Detailed Description

Definition at line 1591 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.101 tflite::BuiltinOptionsTraits< tflite::ReverseSequenceOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ReverseSequenceOptions

8.101.1 Detailed Description

Definition at line 1643 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.102 tflite::BuiltinOptionsTraits< tflite::ReverseV2Options > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ReverseV2Options

8.102.1 Detailed Description

Definition at line 1619 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.103 tflite::BuiltinOptionsTraits< tflite::RNNOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_RNNOptions

8.103.1 Detailed Description

Definition at line 1323 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.104 tflite::BuiltinOptionsTraits< tflite::ScatterNdOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ScatterNdOptions

8.104.1 Detailed Description

Definition at line 1683 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.105 tflite::BuiltinOptionsTraits< tflite::SegmentSumOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SegmentSumOptions

8.105.1 Detailed Description

Definition at line 1695 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.106 tflite::BuiltinOptionsTraits< tflite::SelectOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SelectOptions

8.106.1 Detailed Description

Definition at line 1483 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.107 tflite::BuiltinOptionsTraits< tflite::SelectV2Options > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SelectV2Options

8.107.1 Detailed Description

Definition at line 1687 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.108 tflite::BuiltinOptionsTraits< tflite::SequenceRNNOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SequenceRNNOptions

8.108.1 Detailed Description

Definition at line 1419 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.109 tflite::BuiltinOptionsTraits< tflite::ShapeOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ShapeOptions

8.109.1 Detailed Description

Definition at line 1515 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.110 tflite::BuiltinOptionsTraits< tflite::SkipGramOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SkipGramOptions

8.110.1 Detailed Description

Definition at line 1367 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.111 tflite::BuiltinOptionsTraits< tflite::SliceOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SliceOptions

8.111.1 Detailed Description

Definition at line 1487 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.112 tflite::BuiltinOptionsTraits< tflite::SoftmaxOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SoftmaxOptions

8.112.1 Detailed Description

Definition at line 1331 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.113 tflite::BuiltinOptionsTraits< tflite::SpaceToBatchNDOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SpaceToBatchNDOptions

8.113.1 Detailed Description

Definition at line 1395 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.114 tflite::BuiltinOptionsTraits< tflite::SpaceToDepthOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SpaceToDepthOptions

8.114.1 Detailed Description

Definition at line 1371 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.115 tflite::BuiltinOptionsTraits< tflite::SparseToDenseOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SparseToDenseOptions

8.115.1 Detailed Description

Definition at line 1495 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.116 tflite::BuiltinOptionsTraits< tflite::SplitOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SplitOptions

8.116.1 Detailed Description

Definition at line 1435 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.117 tflite::BuiltinOptionsTraits< tflite::SplitVOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SplitVOptions

8.117.1 Detailed Description

Definition at line 1611 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.118 tflite::BuiltinOptionsTraits< tflite::SquaredDifferenceOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SquaredDifferenceOptions

8.118.1 Detailed Description

Definition at line 1599 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.119 tflite::BuiltinOptionsTraits< tflite::SquareOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SquareOptions

8.119.1 Detailed Description

Definition at line 1559 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.120 tflite::BuiltinOptionsTraits< tflite::SqueezeOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SqueezeOptions

8.120.1 Detailed Description

Definition at line 1415 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.121 tflite::BuiltinOptionsTraits< tflite::StridedSliceOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_StridedSliceOptions

8.121.1 Detailed Description

Definition at line 1423 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.122 tflite::BuiltinOptionsTraits< tflite::SubOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SubOptions

8.122.1 Detailed Description

Definition at line 1407 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.123 tflite::BuiltinOptionsTraits< tflite::SVDFOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_SVDFOptions

8.123.1 Detailed Description

Definition at line 1319 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.124 tflite::BuiltinOptionsTraits< tflite::TileOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_TileOptions

8.124.1 Detailed Description

Definition at line 1499 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.125 tflite::BuiltinOptionsTraits< tflite::TopKV2Options > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_TopKV2Options

8.125.1 Detailed Description

Definition at line 1431 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.126 tflite::BuiltinOptionsTraits< tflite::TransposeConvOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_TransposeConvOptions

8.126.1 Detailed Description

Definition at line 1491 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.127 tflite::BuiltinOptionsTraits< tflite::TransposeOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_TransposeOptions

8.127.1 Detailed Description

Definition at line 1399 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.128 tflite::BuiltinOptionsTraits< tflite::UnidirectionalSequenceLSTMOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_UnidirectionalSequenceLSTMOptions

8.128.1 Detailed Description

Definition at line 1579 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.129 tflite::BuiltinOptionsTraits< tflite::UniqueOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_UniqueOptions

8.129.1 Detailed Description

Definition at line 1615 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.130 tflite::BuiltinOptionsTraits< tflite::UnpackOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_UnpackOptions

8.130.1 Detailed Description

Definition at line 1551 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.131 tflite::BuiltinOptionsTraits< tflite::WhereOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_WhereOptions

8.131.1 Detailed Description

Definition at line 1635 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.132 tflite::BuiltinOptionsTraits< tflite::WhileOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_WhileOptions

8.132.1 Detailed Description

Definition at line 1667 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.133 tflite::BuiltinOptionsTraits< tflite::ZerosLikeOptions > Struct Reference

Static Public Attributes

- static const BuiltinOptions **enum_value** = BuiltinOptions_ZerosLikeOptions

8.133.1 Detailed Description

Definition at line 1563 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.134 nntrainer::CacheElem Class Reference

Cache element containing swap address.

`#include <cache_elem.h>`

Public Types

- enum Options { NONE = 0b0000, FIRST_ACCESS = 0x0001, LAST_ACCESS = 0x0010 }

Public Member Functions

- CacheElem (std::shared_ptr< SwapDevice > dev, unsigned int mem_id, size_t off, size_t len, std::shared_ptr< MemoryData< float >> data, CachePolicy pol=CachePolicy::ALWAYS_SYNCED)

  CacheElem default constructor.
- virtual ~CacheElem ()

  CacheElem destructor.
- void swapIn (Options opt=Options::NONE)

  load data from swap device
- void swapOut (Options opt=Options::NONE)

  unload data to swap device
- bool isActive () const

  unload data to swap device
- size_t getLength () const

  get length of cache element
- unsigned int getId () const

  get id of cache element
- void reset ()

  reset access count

8.134.1 Detailed Description

Cache element containing swap address.

Definition at line 45 of file cache_elem.h.

8.134.2 Member Enumeration Documentation

8.134.2.1 Options

```
enum nntrainer::CacheElem::Options
```

Enumerator

NONE	No option
FIRST_ACCESS	First Access
LAST_ACCESS	Last Access

Definition at line 47 of file cache_elem.h.

8.134.3 Constructor & Destructor Documentation

8.134.3.1 CacheElem()

```
nntrainer::CacheElem::CacheElem (
            std::shared_ptr< SwapDevice > dev,
            unsigned int mem_id,
            size_t off,
            size_t len,
            std::shared_ptr< MemoryData< float >> data,
            CachePolicy pol = CachePolicy::ALWAYS_SYNCED )  [inline], [explicit]
```

CacheElem default constructor.

Definition at line 57 of file cache_elem.h.

8.134.3.2 ~CacheElem()

```
virtual nntrainer::CacheElem::~CacheElem ( )  [inline], [virtual]
```

CacheElem destructor.

Definition at line 74 of file cache_elem.h.

8.134.4 Member Function Documentation

8.134.4.1 getId()

```
unsigned int nntrainer::CacheElem::getId ( ) const  [inline]
```

get id of cache element

Returns

cache element id

Definition at line 109 of file cache_elem.h.

8.134.4.2 getLength()

```
size_t nntrainer::CacheElem::getLength ( ) const  [inline]
```

get length of cache element

Returns

length of cache element in byte

Definition at line 102 of file cache_elem.h.

8.134.4.3 isActive()

```
bool nntrainer::CacheElem::isActive ( ) const  [inline]
```

unload data to swap device

Returns

active status

Definition at line 95 of file cache_elem.h.

8.134.4.4 reset()

```
void nntrainer::CacheElem::reset ( )  [inline]
```

reset access count

Definition at line 115 of file cache_elem.h.

8.134.4.5 swapIn()

```
void nntrainer::CacheElem::swapIn (
            Options opt = Options::NONE )
```

load data from swap device

Parameters

alloc_only	only allocate buffer without reading data

Definition at line 45 of file cache_elem.cpp.

8.134.4.6 swapOut()

```
void nntrainer::CacheElem::swapOut (
            Options opt = Options::NONE )
```

unload data to swap device

Parameters

dealloc_only	only deallocate buffer without writing data

Definition at line 64 of file cache_elem.cpp.

The documentation for this class was generated from the following files:

- tensor/cache_elem.h
- tensor/cache_elem.cpp

8.135 nntrainer::CacheLoader Class Reference

Cache loader from swap device.

`#include <cache_loader.h>`

Public Member Functions

- CacheLoader (std::shared_ptr< CachePool > cache_pool)
  CacheLoader default constructor.
- virtual ~CacheLoader ()
  CacheLoader destructor.
- virtual void init ()
  initialize loader
- virtual void finish ()
  finish loader
- virtual void load (unsigned int order)
  Load cache data with execution order.
- virtual int loadAsync (unsigned int order, TaskExecutor::CompleteCallback callback)
  Load cache data asynchronously with execution order.
- virtual int loadAsync (unsigned int order, TaskExecutor::CompleteCallback callback, long timeout_ms)
  Load cache data asynchronously with execution order.
- virtual int cancelAsync (int id)
  Cancel async task.

8.135.1 Detailed Description

Cache loader from swap device.

Definition at line 32 of file cache_loader.h.

8.135.2 Constructor & Destructor Documentation

8.135.2.1 CacheLoader()

```
nntrainer::CacheLoader::CacheLoader (
            std::shared_ptr< CachePool > cache_pool )  [explicit]
```

CacheLoader default constructor.

Definition at line 28 of file cache_loader.cpp.

8.135.2.2 ~CacheLoader()

```
nntrainer::CacheLoader::~CacheLoader ( )  [virtual]
```

CacheLoader destructor.

Definition at line 32 of file cache_loader.cpp.

8.135.3 Member Function Documentation

8.135.3.1 cancelAsync()

```
int nntrainer::CacheLoader::cancelAsync (
            int id )  [virtual]
```

Cancel async task.

Parameters

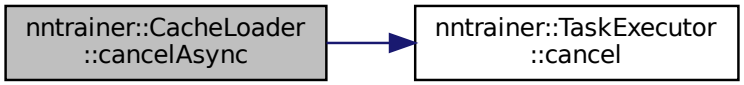
id	task id

Returns

0 on success, otherwise negative error

Definition at line 80 of file cache_loader.cpp.

Here is the call graph for this function:



8.135.3.2 finish()

```
void nntrainer::CacheLoader::finish ( )  [virtual]
```

finish loader

Definition at line 44 of file cache_loader.cpp.

8.135.3.3 load()

```
void nntrainer::CacheLoader::load (
            unsigned int order )  [virtual]
```

Load cache data with execution order.

Parameters

order	execution order

Definition at line 49 of file cache_loader.cpp.

8.135.3.4 loadAsync() [1/2]

```
int nntrainer::CacheLoader::loadAsync (
            unsigned int order,
            TaskExecutor::CompleteCallback callback )  [virtual]
```

Load cache data asynchronously with execution order.

Parameters

order	execution order
complete	complete callback

Returns

async task id

Definition at line 51 of file cache_loader.cpp.

8.135.3.5 loadAsync() [2/2]

```
int nntrainer::CacheLoader::loadAsync (
            unsigned int order,
            TaskExecutor::CompleteCallback callback,
            long timeout_ms ) [virtual]
```

Load cache data asynchronously with execution order.

Parameters

order	execution order
complete	complete callback
timeout	timeout time in ms

Returns

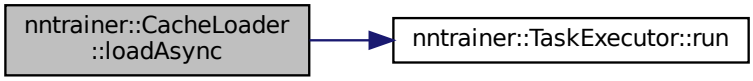
async task id

Note

timeout_ms does not work now.

Definition at line 56 of file cache_loader.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

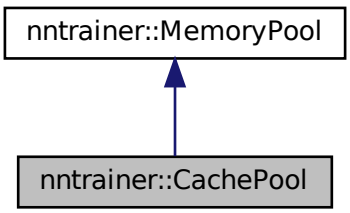
- tensor/cache_loader.h
- tensor/cache_loader.cpp

8.136 nntrainer::CachePool Class Reference

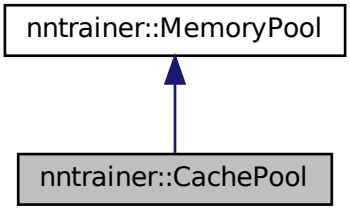
Cache memory with fixed size utilizing swap device.

`#include <cache_pool.h>`

Inheritance diagram for nntrainer::CachePool:



Collaboration diagram for nntrainer::CachePool:



Public Types

- using CacheElems = std::map< unsigned int, std::shared_ptr< CacheElem > >
- using **CacheElemsIter** = CacheElems::iterator
- using **ExecIds** = std::vector< unsigned int >
- using **ExecIdsIter** = ExecIds::iterator

Public Member Functions

- CachePool (const std::string &name)

  CachePool default constructor.
- CachePool (const std::string &path, const std::string &name)

  CachePool constructor with cache path.

- virtual ~CachePool ()

  MemoryPool destructor.
- virtual void allocate ()

  Do the allocation of cache.
- virtual void deallocate ()

  Free all the allocated cache.
- virtual unsigned int requestMemory (size_t bytes, unsigned int start_time, unsigned int end_time, std::vector< unsigned int > exec_order=std::vector< unsigned int >(), TensorLifespan lifespan=TensorLifespan::MAX_LIFESPAN)

  Request Memory from memory pool.
- virtual std::shared_ptr< MemoryData< float > > getMemory (unsigned int id)

  Get the allocated cache.
- virtual bool isAllocated () const

  Is the cache pool allocated.
- virtual void flush ()

  Flush cache data to device.
- virtual void flushExcept (unsigned int order)

  Flush cache data to device except given order.
- virtual void flushExcept (std::vector< unsigned int > order)

  Flush cache data to device except given order.
- virtual void clear ()

  Clear the memory pool.
- virtual void loadExec (unsigned int order)

  Load cache data by execution order.
- virtual void initCacheElemIter (CacheElemsIter &iter)

  Load cache data by execution order.
- virtual bool isLastCacheElemIter (const CacheElemsIter &iter)

  Check iterator is last element.
- virtual void initExecIdsIter (unsigned int order, ExecIdsIter &iter)

  Load cache data by execution order.
- virtual bool isLastExecIdsIter (unsigned int order, const ExecIdsIter &iter)

  Check iterator is last element.
- virtual bool loadExecOnce (unsigned int order, ExecIdsIter &iter)

  Load cache data by execution order.
- virtual void unloadExec (unsigned int order)

  Unload cache data by execution order.
- virtual void loadActives ()

  Load active cache data.
- virtual void unloadActives ()

  Unload active cache data.
- virtual std::string getName ()

  Get name.

Protected Member Functions

- virtual void validate (unsigned int id)

  validate cache element
- virtual void invalidate (unsigned int id)

  invalidate cache element
- std::vector< CachePolicy > & getCachePolicy ()

  Get cache policies.

8.136.1 Detailed Description

Cache memory with fixed size utilizing swap device.

Definition at line 31 of file cache_pool.h.

8.136.2 Member Typedef Documentation

8.136.2.1 CacheElems

```
using nntrainer::CachePool::CacheElems = std::map<unsigned int, std::shared_ptr<CacheElem> >
```

cache id, cache elem

Definition at line 35 of file cache_pool.h.

8.136.3 Constructor & Destructor Documentation

8.136.3.1 CachePool() [1/2]

```
nntrainer::CachePool::CachePool (
            const std::string & name )  [explicit]
```

CachePool default constructor.

Parameters

name	name of the cache pool

Definition at line 94 of file cache_pool.cpp.

8.136.3.2 CachePool() [2/2]

```
nntrainer::CachePool::CachePool (
            const std::string & path,
            const std::string & name )  [explicit]
```

CachePool constructor with cache path.

Definition at line 99 of file cache_pool.cpp.

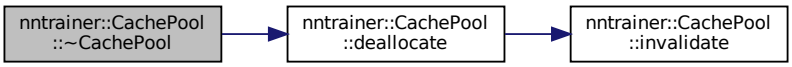
8.136.3.3 ~CachePool()

```
nntrainer::CachePool::~CachePool ( )  [virtual]
```

MemoryPool destructor.

Definition at line 109 of file cache_pool.cpp.

Here is the call graph for this function:



8.136.4 Member Function Documentation

8.136.4.1 allocate()

```
void nntrainer::CachePool::allocate ( )  [virtual]
```

Do the allocation of cache.

Reimplemented from nntrainer::MemoryPool.

Definition at line 117 of file cache_pool.cpp.

8.136.4.2 clear()

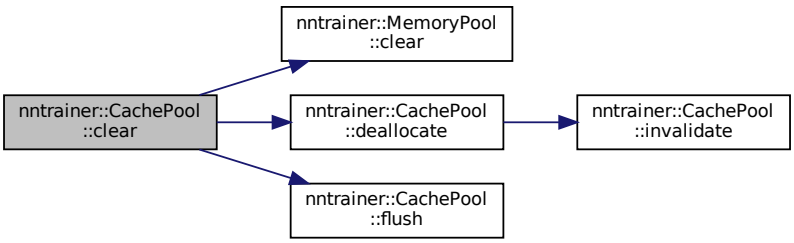
```
void nntrainer::CachePool::clear ( )  [virtual]
```

Clear the memory pool.

Reimplemented from nntrainer::MemoryPool.

Definition at line 257 of file cache_pool.cpp.

Here is the call graph for this function:



8.136.4.3 deallocate()

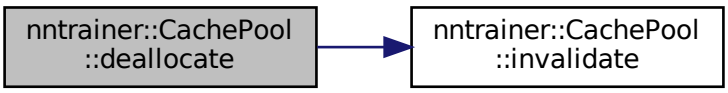
```
void nntrainer::CachePool::deallocate ( )  [virtual]
```

Free all the allocated cache.

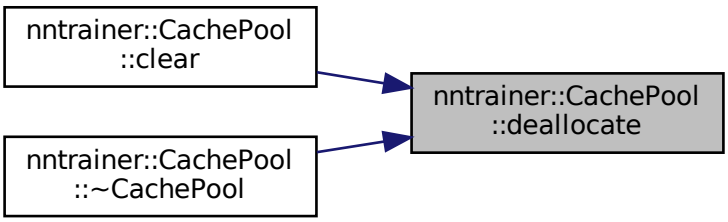
Reimplemented from nntrainer::MemoryPool.

Definition at line 129 of file cache_pool.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.136.4.4 flush()

```
void nntrainer::CachePool::flush ( )  [virtual]
```

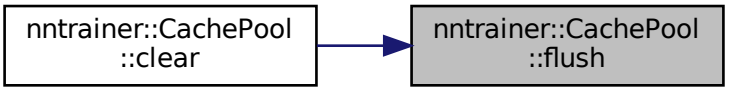
Flush cache data to device.

Note

it must be called only when epoch ends.

Definition at line 198 of file cache_pool.cpp.

Here is the caller graph for this function:



8.136.4.5 flushExcept() [1/2]

```
void nntrainer::CachePool::flushExcept (
            std::vector< unsigned int > order )  [virtual]
```

Flush cache data to device except given order.

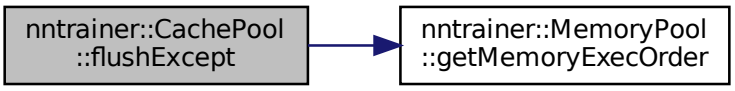
Parameters

order	except execution order

We assumes that flushExcept will be called in front of each execution order, and the order is incremental. So, we can conclude that, if the order passes by the max order of the cache element, it was LAST access of the element.

Definition at line 232 of file cache_pool.cpp.

Here is the call graph for this function:



8.136.4.6 flushExcept() [2/2]

```
void nntrainer::CachePool::flushExcept (
            unsigned int order )  [virtual]
```

Flush cache data to device except given order.

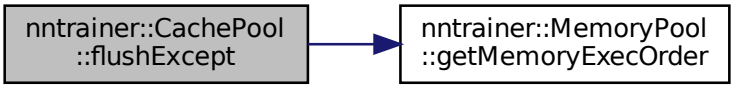
Parameters

order	except execution order

We assumes that flushExcept will be called in front of each execution order, and the order is incremental. So, we can conclude that, if the order passes by the max order of the cache element, it was LAST access of the element.

Definition at line 208 of file cache_pool.cpp.

Here is the call graph for this function:



8.136.4.7 getCachePolicy()

```
std::vector<CachePolicy>& nntrainer::CachePool::getCachePolicy ( )  [inline], [protected]
```

Get cache policies.

Returns

Cache polices

Definition at line 211 of file cache_pool.h.

8.136.4.8 getMemory()

```
std::shared_ptr< MemoryData< float > > nntrainer::CachePool::getMemory (
            unsigned int id )  [virtual]
```

Get the allocated cache.

Parameters

id	The token received from the requestMemory

Returns

The pointer of the cache

This function will throw if called before allocation.

Reimplemented from nntrainer::MemoryPool.

Definition at line 171 of file cache_pool.cpp.

8.136.4.9 getName()

```
virtual std::string nntrainer::CachePool::getName ( )  [inline], [virtual]
```

Get name.

Returns

cache pool name

Definition at line 189 of file cache_pool.h.

8.136.4.10 initCacheElemIter()

```
void nntrainer::CachePool::initCacheElemIter (
            CacheElemsIter & iter )  [virtual]
```

Load cache data by execution order.

Parameters

order	execution order

Definition at line 271 of file cache_pool.cpp.

8.136.4.11 initExecIdsIter()

```
void nntrainer::CachePool::initExecIdsIter (
            unsigned int order,
            ExecIdsIter & iter )  [virtual]
```

Load cache data by execution order.

Parameters

order	execution order

Definition at line 279 of file cache_pool.cpp.

8.136.4.12 invalidate()

```
void nntrainer::CachePool::invalidate (
            unsigned int id )  [protected], [virtual]
```

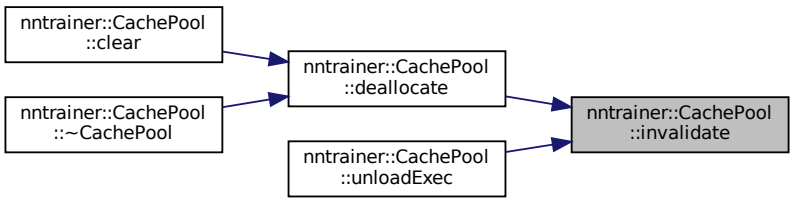
invalidate cache element

Parameters

cache	element id

Definition at line 147 of file cache_pool.cpp.

Here is the caller graph for this function:



8.136.4.13 isAllocated()

```
bool nntrainer::CachePool::isAllocated ( ) const  [virtual]
```

Is the cache pool allocated.

Returns

true if the memory is allocated, else false

Reimplemented from nntrainer::MemoryPool.

Definition at line 264 of file cache_pool.cpp.

8.136.4.14 isLastCacheElemIter()

```
bool nntrainer::CachePool::isLastCacheElemIter (
            const CacheElemsIter & iter )  [virtual]
```

Check iterator is last element.

Parameters

order	execution order

Definition at line 275 of file cache_pool.cpp.

8.136.4.15 isLastExecIdsIter()

```
bool nntrainer::CachePool::isLastExecIdsIter (
            unsigned int order,
            const ExecIdsIter & iter )  [virtual]
```

Check iterator is last element.

Parameters

order	execution order

Definition at line 283 of file cache_pool.cpp.

8.136.4.16 loadExec()

```
void nntrainer::CachePool::loadExec (
            unsigned int order )  [virtual]
```

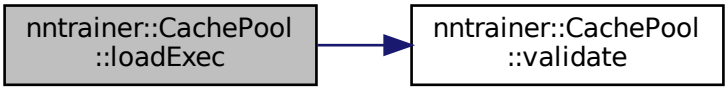
Load cache data by execution order.

Parameters

order	execution order

Definition at line 266 of file cache_pool.cpp.

Here is the call graph for this function:



8.136.4.17 loadExecOnce()

```
bool nntrainer::CachePool::loadExecOnce (
            unsigned int order,
            ExecIdsIter & iter )  [virtual]
```

Load cache data by execution order.

Parameters

order	execution order

Definition at line 287 of file cache_pool.cpp.

Here is the call graph for this function:

8.136.4.18 requestMemory()

```
unsigned int nntrainer::CachePool::requestMemory (
            size_t bytes,
            unsigned int start_time,
            unsigned int end_time,
            std::vector< unsigned int > exec_order = std::vector<unsigned int>(),
            TensorLifespan lifespan = TensorLifespan::MAX_LIFESPAN )  [virtual]
```

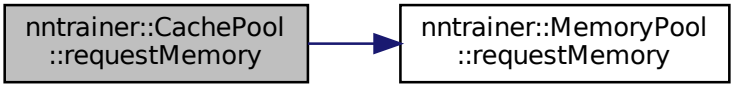
Request Memory from memory pool.

Note

start_time is inclusive, but end_time is exclusive

Definition at line 154 of file cache_pool.cpp.

Here is the call graph for this function:



8.136.4.19 unloadExec()

```
void nntrainer::CachePool::unloadExec (
            unsigned int order )  [virtual]
```

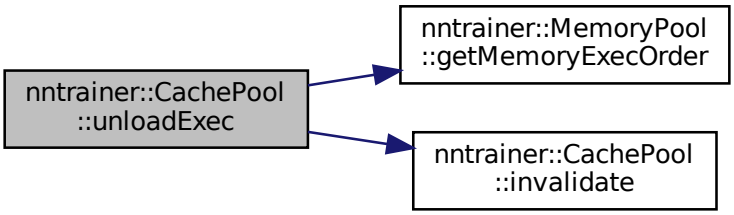
Unload cache data by execution order.

Parameters

order	execution order

Definition at line 297 of file cache_pool.cpp.

Here is the call graph for this function:



8.136.4.20 validate()

```
void nntrainer::CachePool::validate (
            unsigned int id )  [protected], [virtual]
```

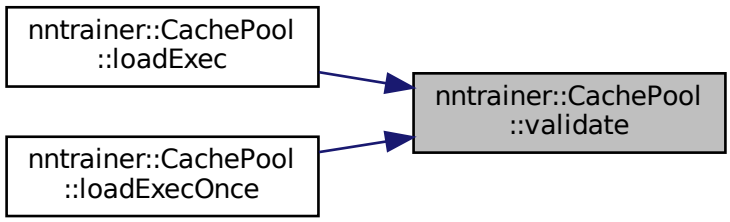
validate cache element

Parameters

cache	element id

Definition at line 140 of file cache_pool.cpp.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- tensor/cache_pool.h
- tensor/cache_pool.cpp

8.137 tflite::CallOptionsBuilder Struct Reference

Public Types

- typedef CallOptions **Table**

Public Member Functions

- void **add_subgraph** (uint32_t subgraph)
- **CallOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- CallOptionsBuilder & **operator=** (const CallOptionsBuilder &)
- flatbuffers::Offset< CallOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.137.1 Detailed Description

Definition at line 4079 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.138 tflite::CastOptionsBuilder Struct Reference

Public Types

- typedef CastOptions **Table**

Public Member Functions

- void **add_in_data_type** (tflite::TensorType in_data_type)
- void **add_out_data_type** (tflite::TensorType out_data_type)
- **CastOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- CastOptionsBuilder & **operator=** (const CastOptionsBuilder &)
- flatbuffers::Offset< CastOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.138.1 Detailed Description

Definition at line 5032 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

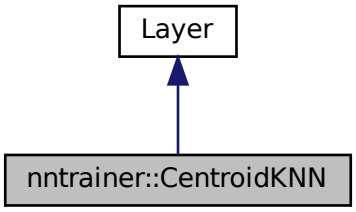
- compiler/tf_schema_generated.h

8.139 nntrainer::CentroidKNN Class Reference

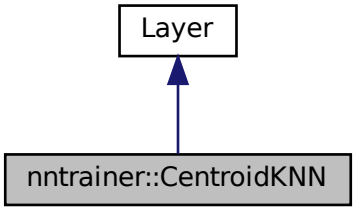
Centroid KNN layer which takes centroid and do k-nearest neighbor classification.

`#include <centroid_knn.h>`

Inheritance diagram for nntrainer::CentroidKNN:



Collaboration diagram for nntrainer::CentroidKNN:



Public Member Functions

- CentroidKNN ()

  Construct a new NearestNeighbors Layer object that does elementwise subtraction from mean feature vector.
- CentroidKNN (CentroidKNN &&rhs) noexcept=default

  Move constructor.
- CentroidKNN & operator= (CentroidKNN &&rhs) noexcept=default

  Move assignment operator. @parma[in] rhs CentroidKNN to be moved.
- ~CentroidKNN ()

  Destroy the NearestNeighbors Layer object.
- bool requireLabel () const override
- void finalize (nntrainer::InitLayerContext &context) override
- void forwarding (nntrainer::RunLayerContext &context, bool training) override
- void calcDerivative (nntrainer::RunLayerContext &context) override

- bool supportBackwarding () const override

  supportBackwarding() const
- void exportTo (nntrainer::Exporter &exporter, const ml::train::ExportMethods &method) const override
- const std::string getType () const override
- void setProperty (const std::vector< std::string > &values) override

Static Public Attributes

- static const std::string **type** = "centroid_knn"

8.139.1 Detailed Description

Centroid KNN layer which takes centroid and do k-nearest neighbor classification.

Definition at line 27 of file centroid_knn.h.

8.139.2 Constructor & Destructor Documentation

8.139.2.1 CentroidKNN()

```
nntrainer::CentroidKNN::CentroidKNN (
            CentroidKNN && rhs )  [default], [noexcept]
```

Move constructor.

Parameters

in	*CentroidKNN*	&&

8.139.2.2 ~CentroidKNN()

```
nntrainer::CentroidKNN::~CentroidKNN ( )
```

Destroy the NearestNeighbors Layer object.

Definition at line 38 of file centroid_knn.cpp.

8.139.3 Member Function Documentation

8.139.3.1 calcDerivative()

```
void nntrainer::CentroidKNN::calcDerivative (
            nntrainer::RunLayerContext & context )  [override]
```

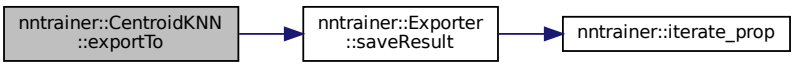
Definition at line 128 of file centroid_knn.cpp.

8.139.3.2 exportTo()

```
void nntrainer::CentroidKNN::exportTo (
            nntrainer::Exporter & exporter,
            const ml::train::ExportMethods & method ) const  [override]
```

Definition at line 133 of file centroid_knn.cpp.

Here is the call graph for this function:



8.139.3.3 finalize()

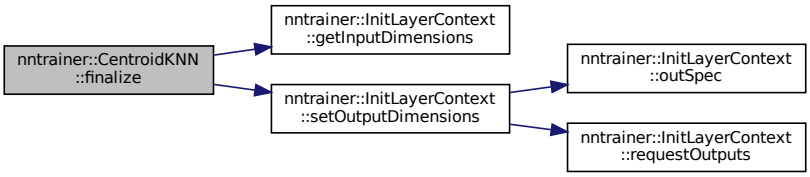
```
void nntrainer::CentroidKNN::finalize (
            nntrainer::InitLayerContext & context )  [override]
```

weight is a distance map that contains centroid of features of each class

samples seen for the current run to calculate the centroid

Definition at line 46 of file centroid_knn.cpp.

Here is the call graph for this function:

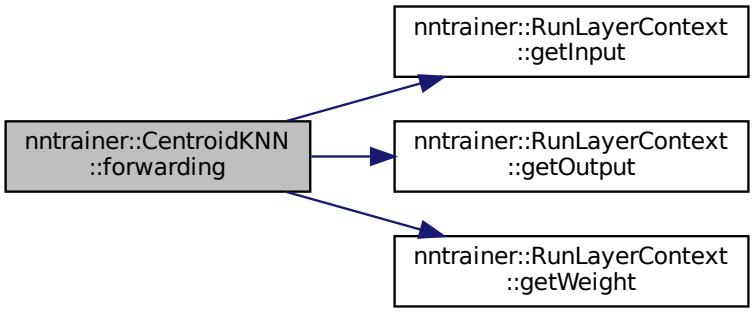


8.139.3.4 forwarding()

```
void nntrainer::CentroidKNN::forwarding (
            nntrainer::RunLayerContext & context,
            bool training )  [override]
```

Definition at line 73 of file centroid_knn.cpp.

Here is the call graph for this function:



8.139.3.5 getType()

```
const std::string nntrainer::CentroidKNN::getType ( ) const  [inline], [override]
```

Definition at line 87 of file centroid_knn.h.

8.139.3.6 requireLabel()

```
bool nntrainer::CentroidKNN::requireLabel ( ) const  [inline], [override]
```

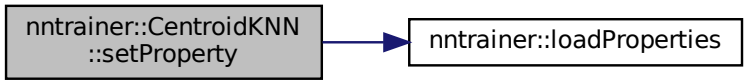
Definition at line 56 of file centroid_knn.h.

8.139.3.7 setProperty()

```
void nntrainer::CentroidKNN::setProperty (
            const std::vector< std::string > & values )  [override]
```

Definition at line 40 of file centroid_knn.cpp.

Here is the call graph for this function:



8.139.3.8 supportBackwarding()

```
bool nntrainer::CentroidKNN::supportBackwarding ( ) const  [inline], [override]
```

supportBackwarding() const

supportBackwarding() const

Definition at line 76 of file centroid_knn.h.

The documentation for this class was generated from the following files:

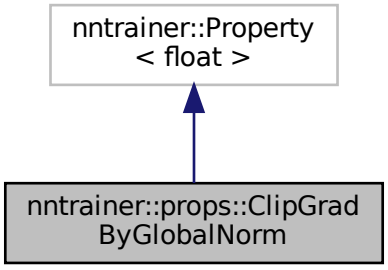
- layers/centroid_knn.h
- layers/centroid_knn.cpp

8.140 nntrainer::props::ClipGradByGlobalNorm Class Reference

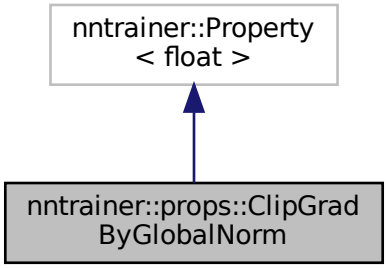
properties for getting the clipping value to clip the gradient by norm

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::ClipGradByGlobalNorm:



Collaboration diagram for nntrainer::props::ClipGradByGlobalNorm:



Public Types

- using prop_tag = float_prop_tag

Static Public Attributes

- static constexpr const char * key

8.140.1 Detailed Description

properties for getting the clipping value to clip the gradient by norm

Definition at line 1273 of file common_properties.h.

8.140.2 Member Typedef Documentation

8.140.2.1 prop_tag

```
using nntrainer::props::ClipGradByGlobalNorm::prop_tag = float_prop_tag
```

property type

Definition at line 1277 of file common_properties.h.

8.140.3 Member Data Documentation

8.140.3.1 key

```
constexpr const char* nntrainer::props::ClipGradByGlobalNorm::key  [static], [constexpr]
```

Initial value:

```
=
    "clip_grad_by_norm"
```

unique key to access

Definition at line 1275 of file common_properties.h.

The documentation for this class was generated from the following file:

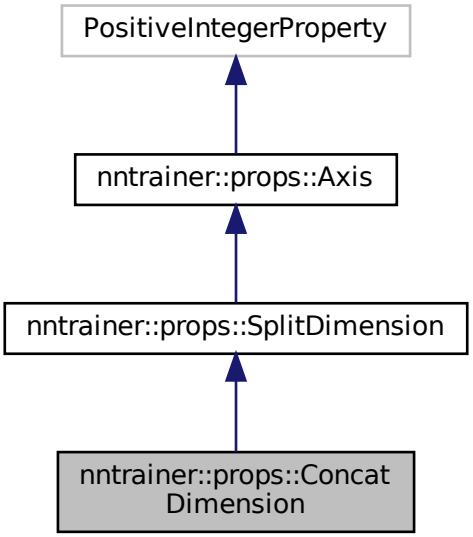
- layers/common_properties.h

8.141 nntrainer::props::ConcatDimension Class Reference

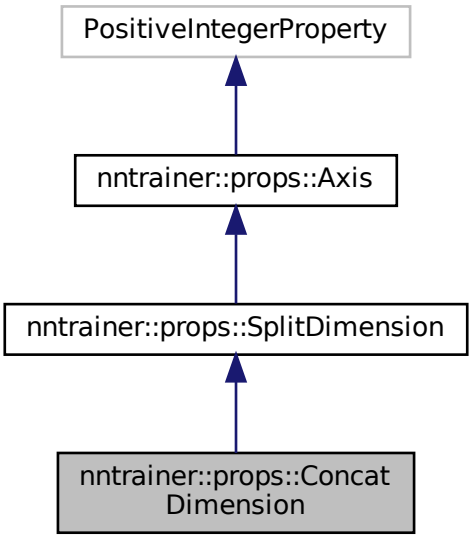
ConcatDimension property, dimension along which to concat the input.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::ConcatDimension:



Collaboration diagram for nntrainer::props::ConcatDimension:



Additional Inherited Members

8.141.1 Detailed Description

ConcatDimension property, dimension along which to concat the input.

Definition at line 308 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.142 tflite::ConcatEmbeddingsOptionsBuilder Struct Reference

Public Types

- typedef ConcatEmbeddingsOptions **Table**

Public Member Functions

- void **add_num_channels** (int32_t num_channels)
- void **add_num_columns_per_channel** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> num_↩ columns_per_channel)
- void **add_embedding_dim_per_channel** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> embedding↩ _dim_per_channel)
- **ConcatEmbeddingsOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ConcatEmbeddingsOptionsBuilder & **operator=** (const ConcatEmbeddingsOptionsBuilder &)
- flatbuffers::Offset< ConcatEmbeddingsOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.142.1 Detailed Description

Definition at line 2989 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.143 tflite::ConcatenationOptionsBuilder Struct Reference

Public Types

- typedef ConcatenationOptions **Table**

Public Member Functions

- void **add_axis** (int32_t axis)
- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- **ConcatenationOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ConcatenationOptionsBuilder & **operator=** (const ConcatenationOptionsBuilder &)
- flatbuffers::Offset< ConcatenationOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.143.1 Detailed Description

Definition at line 3464 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.144 nntrainer::Connection Class Reference

RAII class to define a connection.

`#include <connection.h>`

Public Member Functions

- Connection (const std::string &layer_name, unsigned int idx)

  Construct a new Connection object.
- Connection (const std::string &str_repr)

  Construct a new Connection object from string representation string representation is format of {layer_name, idx};.
- Connection (const Connection &rhs)

  Construct a new Connection object.
- Connection & operator= (const Connection &rhs)

  Copy assignment operator.
- Connection (Connection &&rhs) noexcept

  Move Construct Connection object.
- Connection & operator= (Connection &&rhs) noexcept

  Move assign a connection operator.
- std::string toString () const

  string representation of connection
- const unsigned getIndex () const

  Get the index.
- unsigned & getIndex ()

  Get the index.
- const std::string & getName () const

  Get the Layer name object.
- std::string & getName ()

  Get the Layer name object.
- bool operator== (const Connection &rhs) const noexcept

  operator==

8.144.1 Detailed Description

RAII class to define a connection.

Note

connection is a light weight class wraps around connection information

Definition at line 24 of file connection.h.

8.144.2 Constructor & Destructor Documentation

8.144.2.1 Connection() [1/4]

```
nntrainer::Connection::Connection (
            const std::string & layer_name,
            unsigned int idx )
```

Construct a new Connection object.

Parameters

layer_name	layer identifier
idx	index denotes nth tensor in a layer

Definition at line 20 of file connection.cpp.

8.144.2.2 Connection() [2/4]

```
nntrainer::Connection::Connection (
            const std::string & str_repr )  [explicit]
```

Construct a new Connection object from string representation string representation is format of {layer_name, idx};.

Parameters

str_repr	string format of {layer_name}({idx})

idx_part must not have '(' or ')' inside

Definition at line 24 of file connection.cpp.

8.144.2.3 Connection() [3/4]

```
nntrainer::Connection::Connection (
            const Connection & rhs )  [default]
```

Construct a new Connection object.

Parameters

rhs	rhs to copy

8.144.2.4 Connection() [4/4]

```
nntrainer::Connection::Connection (
            Connection && rhs )  [default], [noexcept]
```

Move Construct Connection object.

Parameters

rhs	rhs to move

8.144.3 Member Function Documentation

8.144.3.1 getIndex() [1/2]

```
unsigned& nntrainer::Connection::getIndex ( )  [inline]
```

Get the index.

Returns

unsigned index

Definition at line 91 of file connection.h.

8.144.3.2 getIndex() [2/2]

```
const unsigned nntrainer::Connection::getIndex ( ) const  [inline]
```

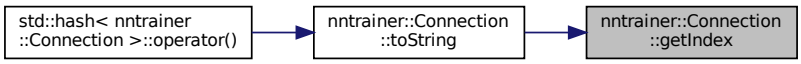
Get the index.

Returns

unsigned index

Definition at line 84 of file connection.h.

Here is the caller graph for this function:



8.144.3.3 getName() [1/2]

```
std::string& nntrainer::Connection::getName ( )  [inline]
```

Get the Layer name object.

Returns

Name& name of layer

Definition at line 105 of file connection.h.

8.144.3.4 getName() [2/2]

```
const std::string& nntrainer::Connection::getName ( ) const  [inline]
```

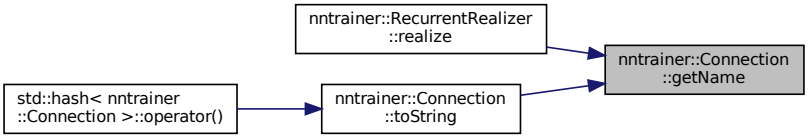
Get the Layer name object.

Returns

const Name& name of layer

Definition at line 98 of file connection.h.

Here is the caller graph for this function:



8.144.3.5 operator=() [1/2]

```
Connection & nntrainer::Connection::operator= (
            Connection && rhs )  [default], [noexcept]
```

Move assign a connection operator.

Parameters

rhs	rhs to move

Returns

Connection&

8.144.3.6 operator=() [2/2]

```
Connection & nntrainer::Connection::operator= (
            const Connection & rhs )  [default]
```

Copy assignment operator.

Parameters

rhs	rhs to copy

Returns

Connection&

8.144.3.7 operator==()

```
bool nntrainer::Connection::operator== (
            const Connection & rhs ) const  [noexcept]
```

operator==

Parameters

rhs	right side to compare

Returns

true if equal

false if not equal

Definition at line 52 of file connection.cpp.

8.144.3.8 toString()

```
std::string nntrainer::Connection::toString ( ) const
```

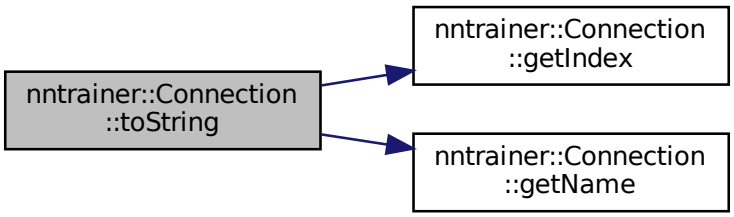
string representation of connection

Returns

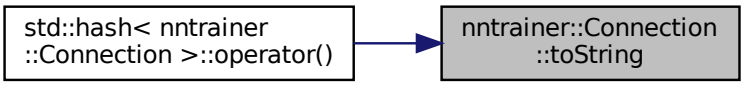
std::string string format of {name}({idx})

Definition at line 56 of file connection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- graph/connection.h
- graph/connection.cpp

8.145 nntrainer::props::connection_prop_tag Struct Reference

Connection prop tag type.

`#include <common_properties.h>`

8.145.1 Detailed Description

Connection prop tag type.

Definition at line 177 of file common_properties.h.

The documentation for this struct was generated from the following file:

- layers/common_properties.h

8.146 tflite::Conv2DOptionsBuilder Struct Reference

Public Types

- typedef Conv2DOptions **Table**

Public Member Functions

- void **add_padding** (tflite::Padding padding)
- void **add_stride_w** (int32_t stride_w)
- void **add_stride_h** (int32_t stride_h)
- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_dilation_w_factor** (int32_t dilation_w_factor)
- void **add_dilation_h_factor** (int32_t dilation_h_factor)
- **Conv2DOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- Conv2DOptionsBuilder & **operator=** (const Conv2DOptionsBuilder &)
- flatbuffers::Offset< Conv2DOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.146.1 Detailed Description

Definition at line 2716 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.147 tflite::CosOptionsBuilder Struct Reference

Public Types

- typedef CosOptions **Table**

Public Member Functions

- **CosOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- CosOptionsBuilder & **operator=** (const CosOptionsBuilder &)
- flatbuffers::Offset< CosOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.147.1 Detailed Description

Definition at line 4700 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.148 tflite::CumsumOptionsBuilder Struct Reference

Public Types

- typedef CumsumOptions **Table**

Public Member Functions

- void **add_exclusive** (bool exclusive)
- void **add_reverse** (bool reverse)
- **CumsumOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- CumsumOptionsBuilder & **operator=** (const CumsumOptionsBuilder &)
- flatbuffers::Offset< CumsumOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.148.1 Detailed Description

Definition at line 7066 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.149 tflite::CustomQuantizationBuilder Struct Reference

Public Types

- typedef CustomQuantization **Table**

Public Member Functions

- void **add_custom** (flatbuffers::Offset< flatbuffers::Vector< uint8_t >> custom)
- **CustomQuantizationBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- CustomQuantizationBuilder & **operator=** (const CustomQuantizationBuilder &)
- flatbuffers::Offset< CustomQuantization > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.149.1 Detailed Description

Definition at line 1981 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

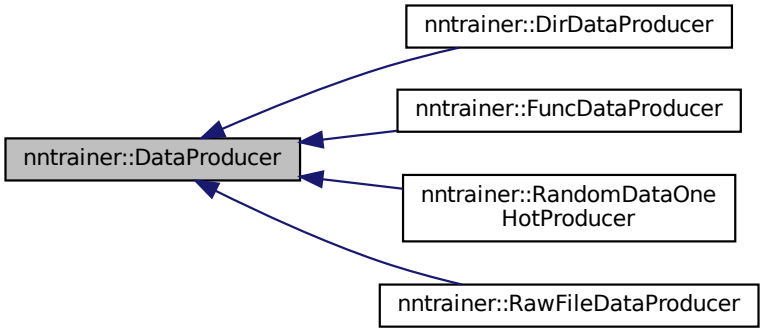
- compiler/tf_schema_generated.h

8.150 nntrainer::DataProducer Class Reference

DataProducer interface used to abstract data provider.

`#include <data_producer.h>`

Inheritance diagram for nntrainer::DataProducer:



Public Types

- using Generator = std::function< bool(unsigned int, std::vector< Tensor > &, std::vector< Tensor > &)>

  generator callable type which will fill a sample

Public Member Functions

- virtual ~DataProducer ()

  Destroy the Data Loader object.
- virtual const std::string getType () const =0

  Get the producer type.
- virtual void setProperty (const std::vector< std::string > &properties)

  Set the Property object.
- virtual Generator finalize (const std::vector< TensorDim > &input_dims, const std::vector< TensorDim > &label_dims, void *user_data=nullptr)

  finalize the class to return an immutable Generator.
- virtual unsigned int size (const std::vector< TensorDim > &input_dims, const std::vector< TensorDim > &label_dims) const

  get the number of samples inside the dataset, if size cannot be determined, this function must return. DataProducer::SIZE_UNDEFINED.
- virtual void exportTo (Exporter &exporter, const ml::train::ExportMethods &method) const

  this function helps exporting the dataproducer in a predefined format, while workarounding issue caused by templated function type eraser
- virtual bool isMultiThreadSafe () const

  denote if given producer is thread safe and can be parallelized.

Static Public Attributes

- constexpr static unsigned int **SIZE_UNDEFINED**

8.150.1 Detailed Description

DataProducer interface used to abstract data provider.

Definition at line 33 of file data_producer.h.

8.150.2 Member Typedef Documentation

8.150.2.1 Generator

```
using nntrainer::DataProducer::Generator = std::function<bool(unsigned int, std::vector<Tensor>
& , std::vector<Tensor> & )>
```

generator callable type which will fill a sample

Parameters

in	*index*	current index with range of [0, size() - 1]. If size() == SIZE_UNDEFINED, this parameter can be ignored
out	*inputs*	allocate tensor before expected to be filled by this function
out	*labels*	allocate tensor before expected to be filled by this function function.

Returns

bool true if this is the last sample, samples will NOT be ignored and should be used, or passed at will of caller

Definition at line 49 of file data_producer.h.

8.150.3 Constructor & Destructor Documentation

8.150.3.1 ~DataProducer()

```
virtual nntrainer::DataProducer::~DataProducer ( )  [inline], [virtual]
```

Destroy the Data Loader object.

Definition at line 58 of file data_producer.h.

8.150.4 Member Function Documentation

8.150.4.1 exportTo()

```
virtual void nntrainer::DataProducer::exportTo (
             Exporter & exporter,
             const ml::train::ExportMethods & method ) const  [inline], [virtual]
```

this function helps exporting the dataproducer in a predefined format, while workarounding issue caused by templated function type eraser

Parameters

exporter	exporter that conatins exporting logic
method	enum value to identify how it should be exported to

Reimplemented in nntrainer::RawFileDataProducer, and nntrainer::FuncDataProducer.

Definition at line 118 of file data_producer.h.

8.150.4.2 finalize()

```
virtual Generator nntrainer::DataProducer::finalize (
            const std::vector< TensorDim > & input_dims,
            const std::vector< TensorDim > & label_dims,
            void * user_data = nullptr )  [inline], [virtual]
```

finalize the class to return an immutable Generator.

Remarks

this function must assume that the batch dimension of each tensor dimension is one. If actual dimension is not one, this function must ignore the batch dimension and assume it to be one.

Parameters

input_dims	input dimensions.
label_dims	label dimensions.
user_data	user data to be used when finalize.

Returns

Generator generator is a function that generates a sample upon call.

Reimplemented in nntrainer::RawFileDataProducer, nntrainer::DirDataProducer, nntrainer::RandomDataOneHotProducer, and nntrainer::FuncDataProducer.

Definition at line 88 of file data_producer.h.

8.150.4.3 getType()

```
virtual const std::string nntrainer::DataProducer::getType ( ) const  [pure virtual]
```

Get the producer type.

Returns

const std::string type representation

Implemented in nntrainer::RawFileDataProducer, nntrainer::DirDataProducer, nntrainer::FuncDataProducer, and nntrainer::RandomDataOneHotProducer.

8.150.4.4 isMultiThreadSafe()

```
virtual bool nntrainer::DataProducer::isMultiThreadSafe ( ) const  [inline], [virtual]
```

denote if given producer is thread safe and can be parallelized.

Note

if size() == SIZE_UNDEFIEND, thread safe shall be false

Returns

bool true if thread safe.

Reimplemented in nntrainer::DirDataProducer, and nntrainer::RandomDataOneHotProducer.

Definition at line 127 of file data_producer.h.

8.150.4.5 setProperty()

```
virtual void nntrainer::DataProducer::setProperty (
            const std::vector< std::string > & properties )  [inline], [virtual]
```

Set the Property object.

Parameters

properties	properties to set

Reimplemented in nntrainer::RawFileDataProducer, nntrainer::DirDataProducer, nntrainer::RandomDataOneHotProducer, and nntrainer::FuncDataProducer.

Definition at line 71 of file data_producer.h.

8.150.4.6 size()

```
virtual unsigned int nntrainer::DataProducer::size (
            const std::vector< TensorDim > & input_dims,
            const std::vector< TensorDim > & label_dims ) const  [inline], [virtual]
```

get the number of samples inside the dataset, if size cannot be determined, this function must return. Data↩ Producer::SIZE_UNDEFINED.

Remarks

this function must assume that the batch dimension of each tensor dimension is one. If actual dimension is not one, this function must ignore the batch dimension and assume it to be one

Parameters

input_dims	input dimensions
label_dims	label dimensions

Returns

size calculated size

Reimplemented in nntrainer::RawFileDataProducer, nntrainer::DirDataProducer, and nntrainer::RandomDataOneHotProducer.

Definition at line 106 of file data_producer.h.

8.150.5 Member Data Documentation

8.150.5.1 SIZE_UNDEFINED

```
constexpr static unsigned int nntrainer::DataProducer::SIZE_UNDEFINED  [inline], [static],
[constexpr]
```

Initial value:

```
=
    std::numeric_limits<unsigned int>::max()
```

Definition at line 51 of file data_producer.h.

The documentation for this class was generated from the following file:

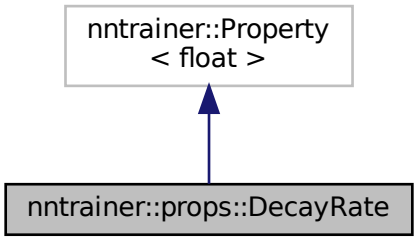
- dataset/data_producer.h

8.151 nntrainer::props::DecayRate Class Reference

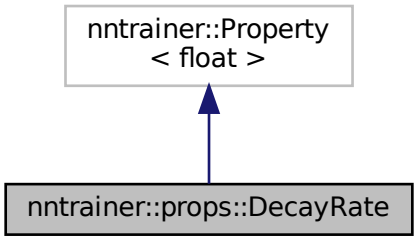
Decay rate property.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::DecayRate:



Collaboration diagram for nntrainer::props::DecayRate:



Public Types

- using prop_tag = float_prop_tag

Static Public Attributes

- static constexpr const char * key = "decay_rate"

8.151.1 Detailed Description

Decay rate property.

Definition at line 1305 of file common_properties.h.

8.151.2 Member Typedef Documentation

8.151.2.1 prop_tag

```
using nntrainer::props::DecayRate::prop_tag = float_prop_tag
```

property type

Definition at line 1308 of file common_properties.h.

8.151.3 Member Data Documentation

8.151.3.1 key

```
constexpr const char* nntrainer::props::DecayRate::key = "decay_rate"  [static], [constexpr]
```

unique key to access

Definition at line 1307 of file common_properties.h.

The documentation for this class was generated from the following file:

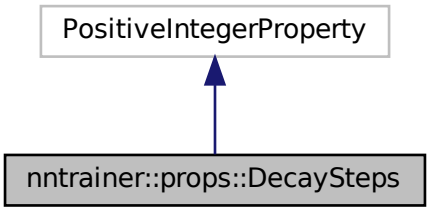
- layers/common_properties.h

8.152 nntrainer::props::DecaySteps Class Reference

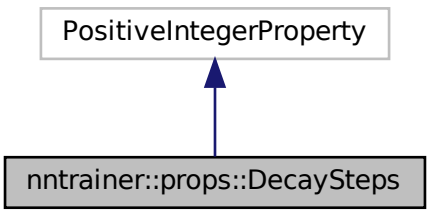
decay steps property

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::DecaySteps:



Collaboration diagram for nntrainer::props::DecaySteps:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key = "decay_steps"

8.152.1 Detailed Description

decay steps property

Definition at line 1315 of file common_properties.h.

8.152.2 Member Typedef Documentation

8.152.2.1 prop_tag

```
using nntrainer::props::DecaySteps::prop_tag = uint_prop_tag
```

property type

Definition at line 1318 of file common_properties.h.

8.152.3 Member Data Documentation

8.152.3.1 key

```
constexpr const char* nntrainer::props::DecaySteps::key = "decay_steps"  [static], [constexpr]
```

unique key to access

Definition at line 1317 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.153 nntrainer::DelegateConfig Class Reference

Configuration for the delegate.

`#include <delegate.h>`

Public Member Functions

- DelegateConfig (Backend backend, Device device)
  Construct a new Delegate Config object.

8.153.1 Detailed Description

Configuration for the delegate.

Definition at line 168 of file delegate.h.

8.153.2 Constructor & Destructor Documentation

8.153.2.1 DelegateConfig()

```
nntrainer::DelegateConfig::DelegateConfig (
            Backend backend,
            Device device )  [inline]
```

Construct a new Delegate Config object.

Parameters

backend	Backend to be set for this delegate
device	Device to be set for the delegate

Definition at line 176 of file delegate.h.

The documentation for this class was generated from the following file:

- delegate.h

8.154 tflite::DensifyOptionsBuilder Struct Reference

Public Types

- typedef DensifyOptions **Table**

Public Member Functions

- **DensifyOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- DensifyOptionsBuilder & **operator=** (const DensifyOptionsBuilder &)
- flatbuffers::Offset< DensifyOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.154.1 Detailed Description

Definition at line 6942 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.155 tflite::DepthToSpaceOptionsBuilder Struct Reference

Public Types

- typedef DepthToSpaceOptions **Table**

Public Member Functions

- void **add_block_size** (int32_t block_size)
- **DepthToSpaceOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- DepthToSpaceOptionsBuilder & **operator=** (const DepthToSpaceOptionsBuilder &)
- flatbuffers::Offset< DepthToSpaceOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.155.1 Detailed Description

Definition at line 4397 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.156 tflite::DepthwiseConv2DOptionsBuilder Struct Reference

Public Types

- typedef DepthwiseConv2DOptions **Table**

Public Member Functions

- void **add_padding** (tflite::Padding padding)
- void **add_stride_w** (int32_t stride_w)
- void **add_stride_h** (int32_t stride_h)
- void **add_depth_multiplier** (int32_t depth_multiplier)
- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_dilation_w_factor** (int32_t dilation_w_factor)
- void **add_dilation_h_factor** (int32_t dilation_h_factor)
- **DepthwiseConv2DOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- DepthwiseConv2DOptionsBuilder & **operator=** (const DepthwiseConv2DOptionsBuilder &)
- flatbuffers::Offset< DepthwiseConv2DOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.156.1 Detailed Description

Definition at line 2905 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.157 tflite::DequantizeOptionsBuilder Struct Reference

Public Types

- typedef DequantizeOptions **Table**

Public Member Functions

- **DequantizeOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- DequantizeOptionsBuilder & **operator=** (const DequantizeOptionsBuilder &)
- flatbuffers::Offset< DequantizeOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.157.1 Detailed Description

Definition at line 5072 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.158 nntrainer::Device Class Reference

Device to be used for the operations to run.

`#include <delegate.h>`

Public Types

- enum DeviceType { DeviceType::cpu, DeviceType::gpu, DeviceType::npu }

  Device type enumeration.

Public Member Functions

- Device (DeviceType device=DeviceType::cpu, float mem_frac=1.0, soft_place=false, prec_loss=false)

  Construct a new Device object.
- void setDevice (DeviceType device)

  Set the Device object.
- DeviceType getDevice ()

  Get the Device object.
- void setMemoryFraction (float memory_fraction)

  Set the maximum memory fraction which can be used by the framework.
- void allowSoftPlacement (bool soft_placement)

  Allow placing the operations on device softly.
- void allowPrecisionLoss (bool precision_loss)

  Allow using low precision version of some of the operations.

8.158.1 Detailed Description

Device to be used for the operations to run.

Note

The selected delegate device will restrict the supported backend.

Definition at line 74 of file delegate.h.

8.158.2 Member Enumeration Documentation

8.158.2.1 DeviceType

```
enum nntrainer::Device::DeviceType  [strong]
```

Device type enumeration.

Enumerator

cpu	CPU as the device
gpu	GPU as the device
npu	NPU as the device

Definition at line 79 of file delegate.h.

8.158.3 Constructor & Destructor Documentation

8.158.3.1 Device()

```
nntrainer::Device::Device (
            DeviceType device = DeviceType::cpu,
            float mem_frac = 1.0,
            soft_place  = false,
            prec_loss  = false )  [inline]
```

Construct a new Device object.

Parameters

device	Device delegate to be used for the operations to run. Default device is cpu.
mem_frac	Maximum memory fraction to be used of the set device.
soft_place	To enable soft device placement.
prec_loss	To enable precision loss for faster computation for the device.

Definition at line 95 of file delegate.h.

8.158.4 Member Function Documentation

8.158.4.1 allowPrecisionLoss()

```
void nntrainer::Device::allowPrecisionLoss (
            bool precision_loss )  [inline]
```

Allow using low precision version of some of the operations.

This allows framework to use low precision operations (like float16 instead of float32) for some of the operations for higher performance with minimum impact to performance

Parameters

precision_loss	True to allow loss in precision, else false

Definition at line 149 of file delegate.h.

8.158.4.2 allowSoftPlacement()

```
void nntrainer::Device::allowSoftPlacement (
            bool soft_placement )  [inline]
```

Allow placing the operations on device softly.

This allows framework to use some other device other than the set device for some of the operations for optimization or if an operation is not supported for the selected device. This might incur extra memory copy overhead.

Note

When set to false, constructing the model can result in error if a required operation is not avaiable for that device.

Parameters

soft_placement	True to allow soft placement, else false

Definition at line 136 of file delegate.h.

8.158.4.3 getDevice()

```
DeviceType nntrainer::Device::getDevice ( )  [inline]
```

Get the Device object.

Returns

DeviceType The device type which is set

Definition at line 114 of file delegate.h.

8.158.4.4 setDevice()

```
void nntrainer::Device::setDevice (
            DeviceType device )  [inline]
```

Set the Device object.

Parameters

device	The device to be set to

Definition at line 107 of file delegate.h.

8.158.4.5 setMemoryFraction()

```
void nntrainer::Device::setMemoryFraction (
            float memory_fraction )  [inline]
```

Set the maximum memory fraction which can be used by the framework.

Parameters

memory_fraction	Maximum fraction of memory to be used

Definition at line 121 of file delegate.h.

The documentation for this class was generated from the following file:

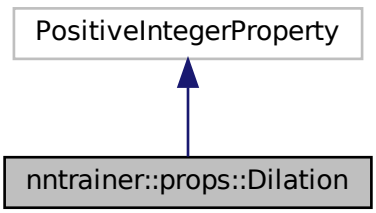
- delegate.h

8.159 nntrainer::props::Dilation Class Reference

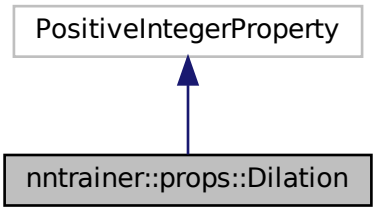
Dilation property, dilation indicates how many space will be inserted between kernel element.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Dilation:



Collaboration diagram for nntrainer::props::Dilation:



Public Types

- using prop_tag = uint_prop_tag

Public Member Functions

- Dilation (unsigned int value=1)

  Construct a new Dilation object with a default value 1.

Static Public Attributes

- static constexpr const char * key = "dilation"

8.159.1 Detailed Description

Dilation property, dilation indicates how many space will be inserted between kernel element.

Definition at line 379 of file common_properties.h.

8.159.2 Member Typedef Documentation

8.159.2.1 prop_tag

```
using nntrainer::props::Dilation::prop_tag = uint_prop_tag
```

property type

Definition at line 387 of file common_properties.h.

8.159.3 Constructor & Destructor Documentation

8.159.3.1 Dilation()

```
nntrainer::props::Dilation::Dilation (
            unsigned int value = 1 )
```

Construct a new Dilation object with a default value 1.

Definition at line 111 of file common_properties.cpp.

8.159.4 Member Data Documentation

8.159.4.1 key

```
constexpr const char* nntrainer::props::Dilation::key = "dilation"  [static], [constexpr]
```

unique key to access

Definition at line 386 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.160 tflite::DimensionMetadataBuilder Struct Reference

Public Types

- typedef DimensionMetadata **Table**

Public Member Functions

- void **add_format** (tflite::DimensionType format)
- void **add_dense_size** (int32_t dense_size)
- void **add_array_segments_type** (tflite::SparseIndexVector array_segments_type)
- void **add_array_segments** (flatbuffers::Offset< void > array_segments)
- void **add_array_indices_type** (tflite::SparseIndexVector array_indices_type)
- void **add_array_indices** (flatbuffers::Offset< void > array_indices)
- **DimensionMetadataBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- DimensionMetadataBuilder & **operator=** (const DimensionMetadataBuilder &)
- flatbuffers::Offset< DimensionMetadata > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.160.1 Detailed Description

Definition at line 2401 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

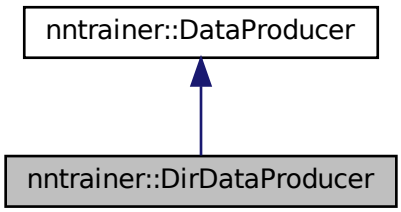
- compiler/tf_schema_generated.h

8.161 nntrainer::DirDataProducer Class Reference

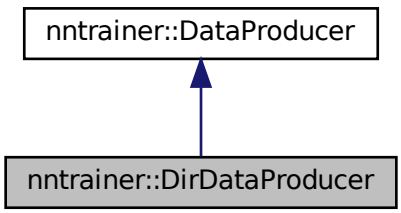
DirDataProducer which generates a onehot vector as a label.

`#include <dir_data_producers.h>`

Inheritance diagram for nntrainer::DirDataProducer:



Collaboration diagram for nntrainer::DirDataProducer:



Public Member Functions

- DirDataProducer ()

  Construct a new Dir Data Producer object.
- DirDataProducer (const std::string &dir_path)

  Construct a new Dir Data Producer object.
- ~DirDataProducer ()

  Destroy the Dir Data Producer object.
- const std::string getType () const override

  Get the producer type.
- bool isMultiThreadSafe () const override

  denote if given producer is thread safe and can be parallelized.
- void setProperty (const std::vector< std::string > &properties) override
- DataProducer::Generator finalize (const std::vector< TensorDim > &input_dims, const std::vector< TensorDim > &label_dims, void *user_data=nullptr) override
- unsigned int size (const std::vector< TensorDim > &input_dims, const std::vector< TensorDim > &label_dims) const override

Static Public Attributes

- static const std::string **type** = "dir"

Additional Inherited Members

8.161.1 Detailed Description

DirDataProducer which generates a onehot vector as a label.

Definition at line 34 of file dir_data_producers.h.

8.161.2 Constructor & Destructor Documentation

8.161.2.1 DirDataProducer() [1/2]

```
nntrainer::DirDataProducer::DirDataProducer ( )
```

Construct a new Dir Data Producer object.

Definition at line 77 of file dir_data_producers.cpp.

8.161.2.2 DirDataProducer() [2/2]

```
nntrainer::DirDataProducer::DirDataProducer (
            const std::string & dir_path )
```

Construct a new Dir Data Producer object.

Definition at line 82 of file dir_data_producers.cpp.

8.161.2.3 ~DirDataProducer()

```
nntrainer::DirDataProducer::~DirDataProducer ( )
```

Destroy the Dir Data Producer object.

Definition at line 87 of file dir_data_producers.cpp.

8.161.3 Member Function Documentation

8.161.3.1 finalize()

```
DataProducer::Generator nntrainer::DirDataProducer::finalize (
            const std::vector< TensorDim > & input_dims,
            const std::vector< TensorDim > & label_dims,
            void * user_data = nullptr )  [override], [virtual]
```

Todo expand this to non onehot case

Reimplemented from nntrainer::DataProducer.

Definition at line 105 of file dir_data_producers.cpp.

8.161.3.2 getType()

```
const std::string nntrainer::DirDataProducer::getType ( ) const  [override], [virtual]
```

Get the producer type.

Returns

const std::string type representation

Implements nntrainer::DataProducer.

Definition at line 89 of file dir_data_producers.cpp.

8.161.3.3 isMultiThreadSafe()

```
bool nntrainer::DirDataProducer::isMultiThreadSafe ( ) const  [override], [virtual]
```

denote if given producer is thread safe and can be parallelized.

Note

if size() == SIZE_UNDEFIEND, thread safe shall be false

Returns

bool true if thread safe.

Todo make this true, it is needed to test multiple worker scenario

Reimplemented from nntrainer::DataProducer.

Definition at line 93 of file dir_data_producers.cpp.

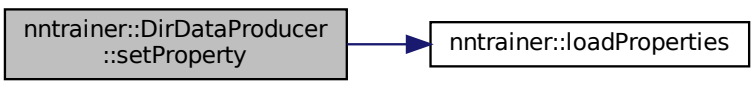
8.161.3.4 setProperty()

```
void nntrainer::DirDataProducer::setProperty (
            const std::vector< std::string > & properties )  [override], [virtual]
```

Reimplemented from nntrainer::DataProducer.

Definition at line 98 of file dir_data_producers.cpp.

Here is the call graph for this function:



8.161.3.5 size()

```
unsigned int nntrainer::DirDataProducer::size (
            const std::vector< TensorDim > & input_dims,
            const std::vector< TensorDim > & label_dims ) const  [override], [virtual]
```

Reimplemented from nntrainer::DataProducer.

Definition at line 169 of file dir_data_producers.cpp.

The documentation for this class was generated from the following files:

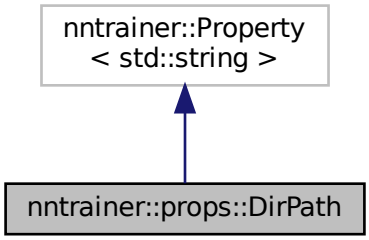
- dataset/dir_data_producers.h
- dataset/dir_data_producers.cpp

8.162 nntrainer::props::DirPath Class Reference

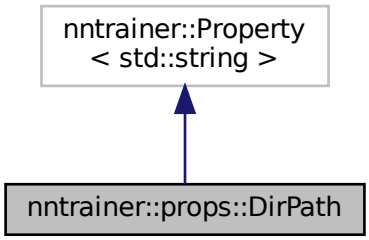
Props containing directory path value.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::DirPath:



Collaboration diagram for nntrainer::props::DirPath:



Public Types

- using prop_tag = str_prop_tag

Public Member Functions

- DirPath ()
  Construct a new Dir Path object.
- DirPath (const std::string &path)
  Construct a new Dir Path object.
- bool isValid (const std::string &v) const override
  check if given value is valid
- void set (const std::string &v) override
  setter

Static Public Attributes

- static constexpr const char * key = "dir_path"

8.162.1 Detailed Description

Props containing directory path value.

Definition at line 596 of file common_properties.h.

8.162.2 Member Typedef Documentation

8.162.2.1 prop_tag

```
using nntrainer::props::DirPath::prop_tag = str_prop_tag
```

property type

Definition at line 610 of file common_properties.h.

8.162.3 Constructor & Destructor Documentation

8.162.3.1 DirPath()

```
nntrainer::props::DirPath::DirPath (
            const std::string & path )  [inline]
```

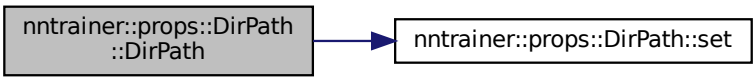
Construct a new Dir Path object.

Parameters

path	path to set

Definition at line 608 of file common_properties.h.

Here is the call graph for this function:



8.162.4 Member Function Documentation

8.162.4.1 isValid()

```
bool nntrainer::props::DirPath::isValid (
            const std::string & v ) const  [override]
```

check if given value is valid

Parameters

v	value to check

Returns

bool true if valid

Definition at line 72 of file common_properties.cpp.

8.162.4.2 set()

```
void nntrainer::props::DirPath::set (
            const std::string & v )  [override]
```

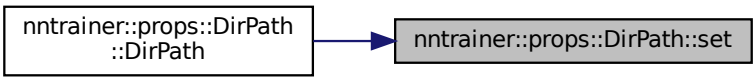
setter

Parameters

v	value to set

Definition at line 77 of file common_properties.cpp.

Here is the caller graph for this function:



8.162.5 Member Data Documentation

8.162.5.1 key

```
constexpr const char* nntrainer::props::DirPath::key = "dir_path"  [static], [constexpr]
```

unique key to access

Definition at line 609 of file common_properties.h.

The documentation for this class was generated from the following files:

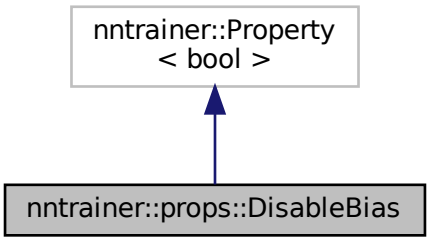
- layers/common_properties.h
- layers/common_properties.cpp

8.163 nntrainer::props::DisableBias Class Reference

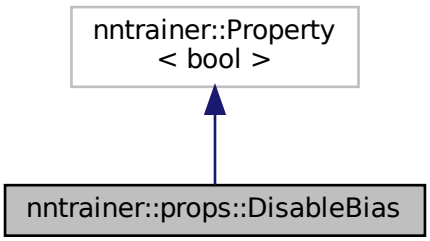
DisableBias to disable the bias.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::DisableBias:



Collaboration diagram for nntrainer::props::DisableBias:



Public Types

- using **prop_tag** = bool_prop_tag

Public Member Functions

- DisableBias (bool val=false)

  Construct a DisableBias object.

Static Public Attributes

- static constexpr const char * **key** = "disable_bias"

8.163.1 Detailed Description

DisableBias to disable the bias.

Definition at line 114 of file common_properties.h.

8.163.2 Constructor & Destructor Documentation

8.163.2.1 DisableBias()

```
nntrainer::props::DisableBias::DisableBias (
            bool val = false )  [inline]
```

Construct a DisableBias object.

Definition at line 120 of file common_properties.h.

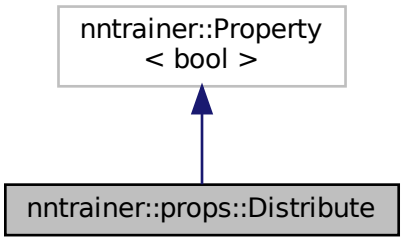
The documentation for this class was generated from the following file:

- layers/common_properties.h

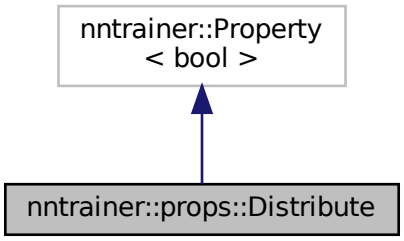
8.164 nntrainer::props::Distribute Class Reference

Distribute property, true if it distribute across layer.

Inheritance diagram for nntrainer::props::Distribute:



Collaboration diagram for nntrainer::props::Distribute:



Public Types

- using **prop_tag** = bool_prop_tag

Static Public Attributes

- static constexpr const char * **key** = "distribute"

8.164.1 Detailed Description

Distribute property, true if it distribute across layer.

Definition at line 58 of file layer_node.cpp.

The documentation for this class was generated from the following file:

- layers/layer_node.cpp

8.165 tflite::DivOptionsBuilder Struct Reference

Public Types

- typedef DivOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- **DivOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- DivOptionsBuilder & **operator=** (const DivOptionsBuilder &)
- flatbuffers::Offset< DivOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.165.1 Detailed Description

Definition at line 4491 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

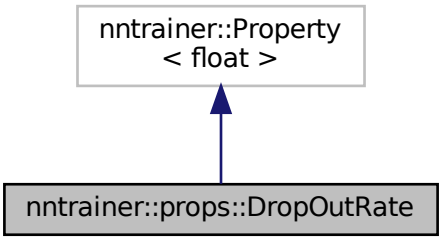
- compiler/tf_schema_generated.h

8.166 nntrainer::props::DropOutRate Class Reference

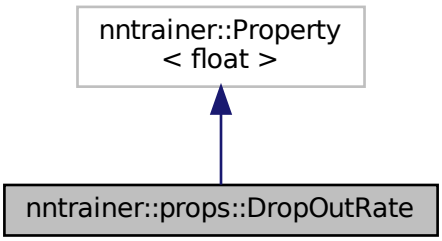
DropOutRate property, this defines drop out specification of layer.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::DropOutRate:



Collaboration diagram for nntrainer::props::DropOutRate:



Public Types

- using prop_tag = float_prop_tag

Public Member Functions

- DropOutRate (float value=0.0)

  Construct a new DropOutRate object with a default value 0.0.
- bool isValid (const float &v) const override

  DropOutRate validator.

Static Public Attributes

- static constexpr const char * key

8.166.1 Detailed Description

DropOutRate property, this defines drop out specification of layer.

Definition at line 504 of file common_properties.h.

8.166.2 Member Typedef Documentation

8.166.2.1 prop_tag

```
using nntrainer::props::DropOutRate::prop_tag = float_prop_tag
```

property type

Definition at line 514 of file common_properties.h.

8.166.3 Constructor & Destructor Documentation

8.166.3.1 DropOutRate()

```
nntrainer::props::DropOutRate::DropOutRate (
            float value = 0.0 )  [inline]
```

Construct a new DropOutRate object with a default value 0.0.

Definition at line 511 of file common_properties.h.

8.166.4 Member Function Documentation

8.166.4.1 isValid()

```
bool nntrainer::props::DropOutRate::isValid (
            const float & v ) const  [override]
```

DropOutRate validator.

Parameters

v	float to validate

Return values

true	if it is greater or equal than 0.0
false	if it is samller than 0.0

Definition at line 53 of file common_properties.cpp.

8.166.5 Member Data Documentation

8.166.5.1 key

```
constexpr const char* nntrainer::props::DropOutRate::key  [static], [constexpr]
```

Initial value:

```
= "dropout_rate"
```

unique key to access

Definition at line 512 of file common_properties.h.

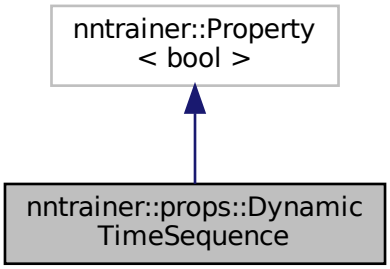
The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

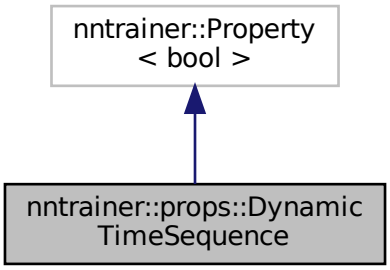
8.167 nntrainer::props::DynamicTimeSequence Class Reference

dynamic time sequence property, use this to set and check if dynamic time sequence is enabled.

Inheritance diagram for nntrainer::props::DynamicTimeSequence:



Collaboration diagram for nntrainer::props::DynamicTimeSequence:



Public Types

- using **prop_tag** = bool_prop_tag

Public Member Functions

- DynamicTimeSequence (bool val=true)

  Construct a new DynamicTimeSequence object.

Static Public Attributes

- static constexpr const char * **key** = "dynamic_time_seq"

8.167.1 Detailed Description

dynamic time sequence property, use this to set and check if dynamic time sequence is enabled.

Definition at line 53 of file recurrent_realizer.cpp.

8.167.2 Constructor & Destructor Documentation

8.167.2.1 DynamicTimeSequence()

```
nntrainer::props::DynamicTimeSequence::DynamicTimeSequence (
            bool val = true ) [inline]
```

Construct a new DynamicTimeSequence object.

Definition at line 59 of file recurrent_realizer.cpp.

The documentation for this class was generated from the following file:

- compiler/recurrent_realizer.cpp

8.168 tflite::EmbeddingLookupSparseOptionsBuilder Struct Reference

Public Types

- typedef EmbeddingLookupSparseOptions **Table**

Public Member Functions

- void **add_combiner** (tflite::CombinerType combiner)
- **EmbeddingLookupSparseOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- EmbeddingLookupSparseOptionsBuilder & **operator=** (const EmbeddingLookupSparseOptionsBuilder &)
- flatbuffers::Offset< EmbeddingLookupSparseOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.168.1 Detailed Description

Definition at line 4563 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

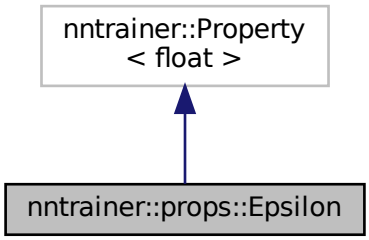
- compiler/tf_schema_generated.h

8.169 nntrainer::props::Epsilon Class Reference

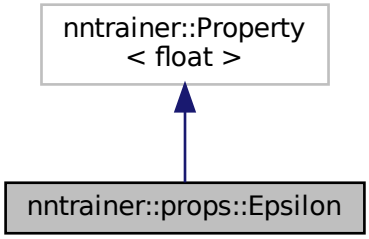
Epsilon property, this is used to avoid divide by zero.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Epsilon:



Collaboration diagram for nntrainer::props::Epsilon:



Public Types

- using prop_tag = float_prop_tag

Public Member Functions

- Epsilon (float value=0.001)

  Construct a new Epsilon object with a default value 0.001.
- bool isValid (const float &value) const override

  Epsilon validator.

Static Public Attributes

- static constexpr const char * key = "epsilon"

8.169.1 Detailed Description

Epsilon property, this is used to avoid divide by zero.

Definition at line 206 of file common_properties.h.

8.169.2 Member Typedef Documentation

8.169.2.1 prop_tag

```
using nntrainer::props::Epsilon::prop_tag = float_prop_tag
```

property type

Definition at line 215 of file common_properties.h.

8.169.3 Constructor & Destructor Documentation

8.169.3.1 Epsilon()

```
nntrainer::props::Epsilon::Epsilon (
            float value = 0.001 )
```

Construct a new Epsilon object with a default value 0.001.

Definition at line 89 of file common_properties.cpp.

8.169.4 Member Function Documentation

8.169.4.1 isValid()

```
bool nntrainer::props::Epsilon::isValid (
            const float & value ) const  [override]
```

Epsilon validator.

Parameters

value	float to validate

Return values

true	if it is greater or equal than 0.0
false	if it is samller than 0.0

Definition at line 91 of file common_properties.cpp.

8.169.5 Member Data Documentation

8.169.5.1 key

```
constexpr const char* nntrainer::props::Epsilon::key = "epsilon"  [static], [constexpr]
```

unique key to access

Definition at line 214 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.170 tflite::EqualOptionsBuilder Struct Reference

Public Types

- typedef EqualOptions **Table**

Public Member Functions

- **EqualOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- EqualOptionsBuilder & **operator=** (const EqualOptionsBuilder &)
- flatbuffers::Offset< EqualOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.170.1 Detailed Description

Definition at line 5590 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.171 nntrainer::exception::ErrorNotification< Err, > Class Template Reference

Error Notification class, error is thrown when the class is destroyed. DO NOT use this outside as this contains throwing destructor.

`#include <nntrainer_error.h>`

Public Member Functions

- ErrorNotification ()

  Construct a new Error Notification object.
- ErrorNotification (std::function< void()> cleanup_func_)

  Construct a new Error Notification object.
- ~ErrorNotification () noexcept(false)

  Destroy the Error Notification object, Error is thrown when destroying this.

Friends

- template<typename T >
  ErrorNotification< Err > && operator<< (ErrorNotification< Err > &&out, T &&e)

  Error notification stream wrapper.

8.171.1 Detailed Description

template<typename Err, typename std::enable_if_t< std::is_base_of< std::exception, Err >::value, Err > * = nullptr>
class nntrainer::exception::ErrorNotification< Err, >

Error Notification class, error is thrown when the class is destroyed. DO NOT use this outside as this contains throwing destructor.

Template Parameters

Err	Error type that except cstring as an argument.

Definition at line 83 of file nntrainer_error.h.

8.171.2 Constructor & Destructor Documentation

8.171.2.1 ErrorNotification() [1/2]

```
template<typename Err , typename std::enable_if_t< std::is_base_of< std::exception, Err >↩
::value, Err > * = nullptr>
nntrainer::exception::ErrorNotification< Err, >::ErrorNotification ( )  [inline], [explicit]
```

Construct a new Error Notification object.

Definition at line 89 of file nntrainer_error.h.

8.171.2.2 ErrorNotification() [2/2]

```
template<typename Err , typename std::enable_if_t< std::is_base_of< std::exception, Err >↩
::value, Err > * = nullptr>
nntrainer::exception::ErrorNotification< Err, >::ErrorNotification (
            std::function< void()> cleanup_func_ )  [inline], [explicit]
```

Construct a new Error Notification object.

Parameters

*cleanup_↩ func_*	clean up function

Definition at line 96 of file nntrainer_error.h.

8.171.2.3 ~ErrorNotification()

```
template<typename Err , typename std::enable_if_t< std::is_base_of< std::exception, Err >↩
::value, Err > * = nullptr>
nntrainer::exception::ErrorNotification< Err, >::~ErrorNotification ( )  [inline], [noexcept]
```

Destroy the Error Notification object, Error is thrown when destroying this.

Definition at line 104 of file nntrainer_error.h.

8.171.3 Friends And Related Function Documentation

8.171.3.1 operator<<

```
template<typename Err , typename std::enable_if_t< std::is_base_of< std::exception, Err >↩
::value, Err > * = nullptr>
template<typename T >
ErrorNotification<Err>&& operator<< (
            ErrorNotification< Err > && out,
            T && e )  [friend]
```

Error notification stream wrapper.

Template Parameters

T	anything that will be delegated to the move

Parameters

out	out stream
e	anything to pass to the stream

Returns

ErrorNotification<Err>&& self

Definition at line 120 of file nntrainer_error.h.

The documentation for this class was generated from the following file:

- nntrainer_error.h

8.172 tflite::ExpandDimsOptionsBuilder Struct Reference

Public Types

- typedef ExpandDimsOptions **Table**

Public Member Functions

- **ExpandDimsOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ExpandDimsOptionsBuilder & **operator=** (const ExpandDimsOptionsBuilder &)
- flatbuffers::Offset< ExpandDimsOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.172.1 Detailed Description

Definition at line 5518 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.173 tflite::ExpOptionsBuilder Struct Reference

Public Types

- typedef ExpOptions **Table**

Public Member Functions

- **ExpOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ExpOptionsBuilder & **operator=** (const ExpOptionsBuilder &)
- flatbuffers::Offset< ExpOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.173.1 Detailed Description

Definition at line 4670 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.174 nntrainer::Exporter Class Reference

Exporter class helps to exports the node information in a predefined way. because each method will require complete different methods, this class exploits visitor pattern to make a custom defined saving method.

`#include <node_exporter.h>`

Public Member Functions

- Exporter ()

  Construct a new Exporter object.
- ~Exporter ()

  Destroy the Exporter object.
- template<typename... Ts, typename NodeType = void>
  void saveResult (const std::tuple< Ts... > &props, ml::train::ExportMethods method, const NodeType *self=nullptr)

  this function iterates over the property and process the property in a designated way.
- template<ml::train::ExportMethods methods, typename T = typename return_type<methods>::type>
  std::unique_ptr< T > getResult ()

  Get the result object.

8.174.1 Detailed Description

Exporter class helps to exports the node information in a predefined way. because each method will require complete different methods, this class exploits visitor pattern to make a custom defined saving method.

Definition at line 89 of file node_exporter.h.

8.174.2 Constructor & Destructor Documentation

8.174.2.1 Exporter()

```
nntrainer::Exporter::Exporter ( )
```

Construct a new Exporter object.

Definition at line 46 of file node_exporter.cpp.

8.174.2.2 ~Exporter()

```
nntrainer::Exporter::~Exporter ( )  [default]
```

Destroy the Exporter object.

8.174.3 Member Function Documentation

8.174.3.1 getResult()

```
template<ml::train::ExportMethods methods, typename T = typename return_type<methods>::type>
std::unique_ptr<T> nntrainer::Exporter::getResult ( )
```

Get the result object.

Note

unique_ptr here will be a member of *this*. The ownership is passed to the caller, thus after getResult, the target member is cleared. This is intended design choice to mimic single function call, between *saveResult* and *getResult*

Template Parameters

methods	method to get
T	appropriate return type regarding the export method

Returns

std::unique_ptr<T> predefined return type according to the method.

Return values

nullptr	not exported

Here is the caller graph for this function:

8.174.3.2 saveResult()

```
template<typename...  Ts, typename NodeType = void>
void nntrainer::Exporter::saveResult (
             const std::tuple< Ts...  > & props,
             ml::train::ExportMethods method,
             const NodeType * self = nullptr )  [inline]
```

this function iterates over the property and process the property in a designated way.

Template Parameters

Ts	type of elements
NodeType	NodeType element

Parameters

props	tuple that contains properties
method	method to export
self	this pointer to the layer which is being exported

function to pass to the iterate_prop, this saves the property to stored_result

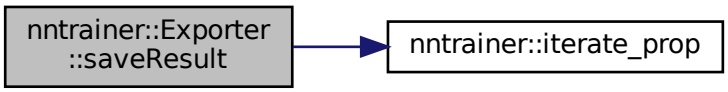
Parameters

prop	property property to pass
index	index of the current property

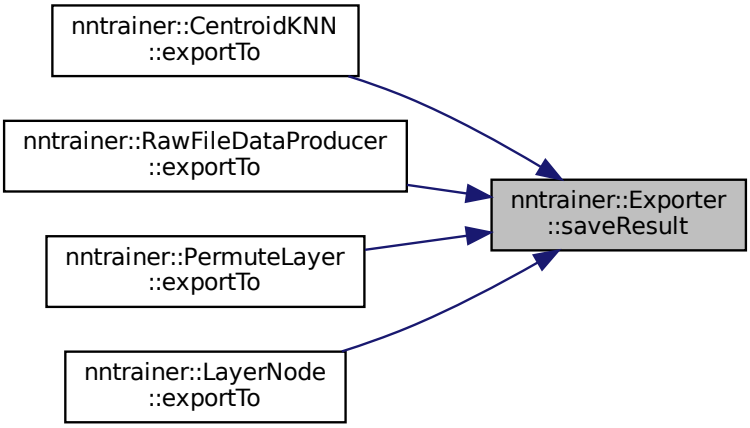
fall through intended

Definition at line 124 of file node_exporter.h.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- utils/node_exporter.h
- utils/node_exporter.cpp

8.175 External Struct Reference

External Loop Info for broadcasted iteration. Please refer to DISABLED_private_external_loop_n in unittest_↩ nntrainer_tensor.

8.175.1 Detailed Description

External Loop Info for broadcasted iteration. Please refer to DISABLED_private_external_loop_n in unittest_↩ nntrainer_tensor.

for broadcasted info

Note

This should better be implemented in iterator fashion before used extensively.

The documentation for this struct was generated from the following file:

- tensor/tensor.cpp

8.176 tflite::FakeQuantOptionsBuilder Struct Reference

Public Types

- typedef FakeQuantOptions **Table**

Public Member Functions

- void **add_min** (float min)
- void **add_max** (float max)
- void **add_num_bits** (int32_t num_bits)
- void **add_narrow_range** (bool narrow_range)
- **FakeQuantOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- FakeQuantOptionsBuilder & **operator=** (const FakeQuantOptionsBuilder &)
- flatbuffers::Offset< FakeQuantOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.176.1 Detailed Description

Definition at line 5774 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

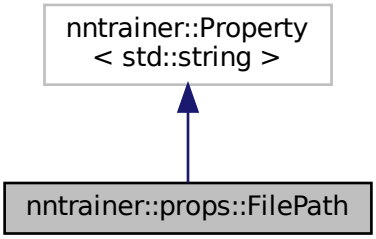
- compiler/tf_schema_generated.h

8.177 nntrainer::props::FilePath Class Reference

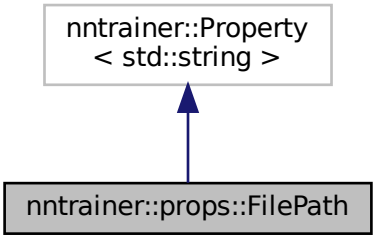
Props containing file path value.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::FilePath:



Collaboration diagram for nntrainer::props::FilePath:



Public Types

- using prop_tag = str_prop_tag

Public Member Functions

- FilePath ()
  Construct a new File Path object.
- FilePath (const std::string &path)
  Construct a new File Path object.
- bool isValid (const std::string &v) const override
  check if given value is valid
- void set (const std::string &v) override
  setter
- std::ifstream::pos_type file_size ()
  return file size

Static Public Attributes

- static constexpr const char * key = "path"

8.177.1 Detailed Description

Props containing file path value.

Definition at line 550 of file common_properties.h.

8.177.2 Member Typedef Documentation

8.177.2.1 prop_tag

```
using nntrainer::props::FilePath::prop_tag = str_prop_tag
```

property type

Definition at line 564 of file common_properties.h.

8.177.3 Constructor & Destructor Documentation

8.177.3.1 FilePath()

```
nntrainer::props::FilePath::FilePath (
            const std::string & path )  [inline]
```

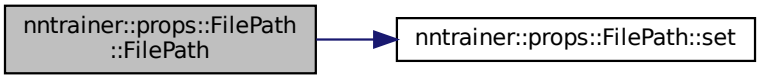
Construct a new File Path object.

Parameters

path	path to set

Definition at line 562 of file common_properties.h.

Here is the call graph for this function:



8.177.4 Member Function Documentation

8.177.4.1 file_size()

```
std::ifstream::pos_type nntrainer::props::FilePath::file_size ( )
```

return file size

Returns

std::ifstream::pos_type size of the file

Definition at line 70 of file common_properties.cpp.

8.177.4.2 isValid()

```
bool nntrainer::props::FilePath::isValid (
            const std::string & v ) const  [override]
```

check if given value is valid

Parameters

v	value to check

Returns

bool true if valid

Definition at line 59 of file common_properties.cpp.

8.177.4.3 set()

```
void nntrainer::props::FilePath::set (
            const std::string & v )  [override]
```

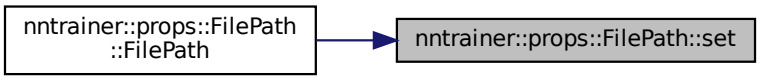
setter

Parameters

v	value to set

Definition at line 64 of file common_properties.cpp.

Here is the caller graph for this function:



8.177.5 Member Data Documentation

8.177.5.1 key

```
constexpr const char* nntrainer::props::FilePath::key = "path"  [static], [constexpr]
```

unique key to access

Definition at line 563 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.178 tflite::FillOptionsBuilder Struct Reference

Public Types

- typedef FillOptions **Table**

Public Member Functions

- **FillOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- FillOptionsBuilder & **operator=** (const FillOptionsBuilder &)
- flatbuffers::Offset< FillOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.178.1 Detailed Description

Definition at line 6210 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

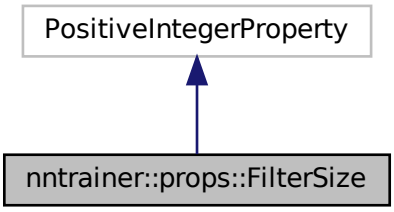
- compiler/tf_schema_generated.h

8.179 nntrainer::props::FilterSize Class Reference

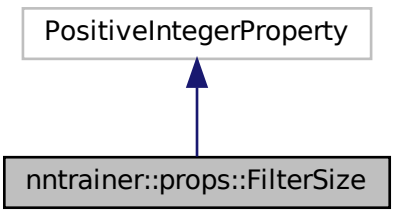
FilterSize property, filter size is used to measure how many filters are there.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::FilterSize:



Collaboration diagram for nntrainer::props::FilterSize:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key = "filters"

8.179.1 Detailed Description

FilterSize property, filter size is used to measure how many filters are there.

Definition at line 321 of file common_properties.h.

8.179.2 Member Typedef Documentation

8.179.2.1 prop_tag

```
using nntrainer::props::FilterSize::prop_tag = uint_prop_tag
```

property type

Definition at line 324 of file common_properties.h.

8.179.3 Member Data Documentation

8.179.3.1 key

```
constexpr const char* nntrainer::props::FilterSize::key = "filters"  [static], [constexpr]
```

unique key to access

Definition at line 323 of file common_properties.h.

The documentation for this class was generated from the following file:

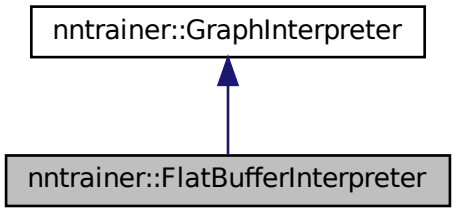
- layers/common_properties.h

8.180 nntrainer::FlatBufferInterpreter Class Reference

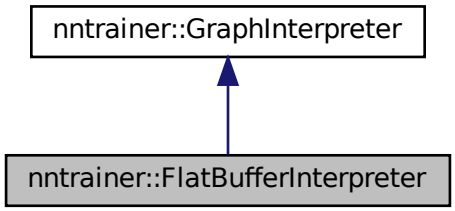
flatbuffer graph interpreter class

`#include <flatbuffer_interpreter.h>`

Inheritance diagram for nntrainer::FlatBufferInterpreter:



Collaboration diagram for nntrainer::FlatBufferInterpreter:



Public Member Functions

- FlatBufferInterpreter (AppContext &app_context_=AppContext::Global())

  Construct a new flatbuffer Graph Interpreter object.
- virtual ~FlatBufferInterpreter ()=default

  Destroy the flatbuffer Interpreter object.
- void serialize (const GraphRepresentation &representation, const std::string &out) override
- GraphRepresentation deserialize (const std::string &in) override

  deserialize graph from a stream

8.180.1 Detailed Description

flatbuffer graph interpreter class

Definition at line 25 of file flatbuffer_interpreter.h.

8.180.2 Constructor & Destructor Documentation

8.180.2.1 FlatBufferInterpreter()

```
nntrainer::FlatBufferInterpreter::FlatBufferInterpreter (
            AppContext & app_context_ = AppContext::Global() )  [inline]
```

Construct a new flatbuffer Graph Interpreter object.

Parameters

*app_↩ context_*	app context to create layers

Definition at line 32 of file flatbuffer_interpreter.h.

8.180.2.2 ~FlatBufferInterpreter()

```
virtual nntrainer::FlatBufferInterpreter::~FlatBufferInterpreter ( )  [virtual], [default]
```

Destroy the flatbuffer Interpreter object.

8.180.3 Member Function Documentation

8.180.3.1 deserialize()

```
GraphRepresentation nntrainer::FlatBufferInterpreter::deserialize (
            const std::string & in )  [override], [virtual]
```

deserialize graph from a stream

Parameters

in	input file name

Returns

GraphRepresentation graph representation

Implements nntrainer::GraphInterpreter.

8.180.3.2 serialize()

```
void nntrainer::FlatBufferInterpreter::serialize (
            const GraphRepresentation & representation,
            const std::string & out )  [override], [virtual]
```

Implements nntrainer::GraphInterpreter.

The documentation for this class was generated from the following file:

- compiler/flatbuffer_interpreter.h

8.181 nntrainer::FlatBufferOpNode Class Reference

FlatBufferOpNode class.

```
#include <flatbuffer_opnode.h>
```

Public Types

- using **Variables** = std::vector< const Tensor * >

Public Member Functions

- FlatBufferOpNode ()
  Construct a new Flat Buffer Op Node object.
- void setLayerNode (const LayerNode &layer)
  Set the Layer Node object.
- void setOpType (nntr::BuiltinOperator op_type_)
  Set the Op Type object.
- void setBuiltinOptions (nntr::BuiltinOptions builtin_option_type_, const flatbuffers::Offset< void > &builtin_ops_)
  Set the Builtin Options object.
- Variables & getInputs ()
  Get the Inputs object.
- const Variables & getInputs () const
  Get the Inputs object.
- Variables & getWeights ()
  Get the Weights object.
- const Variables & getWeights () const
  Get the Weights object.
- Variables & getOutputs ()
  Get the Outputs object.
- const Variables & getOutputs () const
  Get the Outputs object.
- bool isInputNode () const
  check if the node is model input
- bool isOutputNode () const

check if the node is model output

- bool isVirtualNode () const

  check if the node is virtual node

- const nntr::BuiltinOperator getOpType () const

  Get the Op Type object.

- const nntr::BuiltinOptions getOptionType () const

  Get the Option Type object.

- flatbuffers::Offset< void > getBuiltinOps () const

  Get the Builtin Ops object.

- const std::vector< FlatBufferOpNode * > & getInputNodes () const

  Get the Input Nodes object.

- void arity (size_t value)

  Set arity.

- const unsigned arity () const

  Get arity.

- void setArg (size_t index, FlatBufferOpNode *node)

  Set the Arg object.

- FlatBufferOpNode * arg (size_t index) const

  Get the Arg object.

8.181.1 Detailed Description

FlatBufferOpNode class.

Definition at line 32 of file flatbuffer_opnode.h.

8.181.2 Constructor & Destructor Documentation

8.181.2.1 FlatBufferOpNode()

```
nntrainer::FlatBufferOpNode::FlatBufferOpNode ( )
```

Construct a new Flat Buffer Op Node object.

Definition at line 21 of file flatbuffer_opnode.cpp.

8.181.3 Member Function Documentation

8.181.3.1 arg()

```
FlatBufferOpNode* nntrainer::FlatBufferOpNode::arg (
            size_t index ) const  [inline]
```

Get the Arg object.

Parameters

index	argument index to get

Returns

FlatBufferOpNode *input_nodes.at(index)

Definition at line 194 of file flatbuffer_opnode.h.

8.181.3.2 arity() [1/2]

```
const unsigned nntrainer::FlatBufferOpNode::arity ( ) const  [inline]
```

Get arity.

Returns

const unsigned input_nodes size

Definition at line 176 of file flatbuffer_opnode.h.

8.181.3.3 arity() [2/2]

```
void nntrainer::FlatBufferOpNode::arity (
            size_t value )  [inline]
```

Set arity.

Parameters

value	value to set

Definition at line 169 of file flatbuffer_opnode.h.

8.181.3.4 getBuiltinOps()

```
flatbuffers::Offset< void > nntrainer::FlatBufferOpNode::getBuiltinOps ( ) const
```

Get the Builtin Ops object.

Parameters

f	Flatbuffer builder

Returns

flatbuffers::Offset<void>

Definition at line 72 of file flatbuffer_opnode.cpp.

8.181.3.5 getInputNodes()

```
const std::vector<FlatBufferOpNode *>& nntrainer::FlatBufferOpNode::getInputNodes ( ) const
[inline]
```

Get the Input Nodes object.

Returns

const std::vector<FlatBufferOpNode *> &input_nodes

Definition at line 160 of file flatbuffer_opnode.h.

8.181.3.6 getInputs() [1/2]

```
Variables& nntrainer::FlatBufferOpNode::getInputs ( )  [inline]
```

Get the Inputs object.

Returns

Variables& inputs

Definition at line 70 of file flatbuffer_opnode.h.

8.181.3.7 getInputs() [2/2]

```
const Variables& nntrainer::FlatBufferOpNode::getInputs ( ) const  [inline]
```

Get the Inputs object.

Returns

const Variables& inputs

Definition at line 77 of file flatbuffer_opnode.h.

8.181.3.8 getOptionType()

```
const nntr::BuiltinOptions nntrainer::FlatBufferOpNode::getOptionType ( ) const  [inline]
```

Get the Option Type object.

Returns

const nntr::BuiltinOptions

Definition at line 143 of file flatbuffer_opnode.h.

8.181.3.9 getOpType()

```
const nntr::BuiltinOperator nntrainer::FlatBufferOpNode::getOpType ( ) const  [inline]
```

Get the Op Type object.

Returns

const nntr::BuiltinOperator

Definition at line 136 of file flatbuffer_opnode.h.

8.181.3.10 getOutputs() [1/2]

```
Variables& nntrainer::FlatBufferOpNode::getOutputs ( )  [inline]
```

Get the Outputs object.

Returns

Variables& outputs

Definition at line 98 of file flatbuffer_opnode.h.

8.181.3.11 getOutputs() [2/2]

```
const Variables& nntrainer::FlatBufferOpNode::getOutputs ( ) const  [inline]
```

Get the Outputs object.

Returns

const Variables& outputs

Definition at line 105 of file flatbuffer_opnode.h.

8.181.3.12 getWeights() [1/2]

```
Variables& nntrainer::FlatBufferOpNode::getWeights ( )  [inline]
```

Get the Weights object.

Returns

Variables& weights

Definition at line 84 of file flatbuffer_opnode.h.

8.181.3.13 getWeights() [2/2]

```
const Variables& nntrainer::FlatBufferOpNode::getWeights ( ) const  [inline]
```

Get the Weights object.

Returns

const Variables& weights

Definition at line 91 of file flatbuffer_opnode.h.

8.181.3.14 isInputNode()

```
bool nntrainer::FlatBufferOpNode::isInputNode ( ) const  [inline]
```

check if the node is model input

Returns

true if op node is model input
false if op node is not model input

Definition at line 113 of file flatbuffer_opnode.h.

8.181.3.15 isOutputNode()

```
bool nntrainer::FlatBufferOpNode::isOutputNode ( ) const  [inline]
```

check if the node is model output

Returns

true if op node is model output
false if op node is not model output

Definition at line 121 of file flatbuffer_opnode.h.

8.181.3.16 isVirtualNode()

```
bool nntrainer::FlatBufferOpNode::isVirtualNode ( ) const  [inline]
```

check if the node is virtual node

Returns

true if this op node is virtual node

false if this op node is not virtual node

Definition at line 129 of file flatbuffer_opnode.h.

8.181.3.17 setArg()

```
void nntrainer::FlatBufferOpNode::setArg (
            size_t index,
            FlatBufferOpNode * node )  [inline]
```

Set the Arg object.

Parameters

index	argument index to set
node	the node to be argument

Definition at line 184 of file flatbuffer_opnode.h.

8.181.3.18 setBuiltinOptions()

```
void nntrainer::FlatBufferOpNode::setBuiltinOptions (
            nntr::BuiltinOptions builtin_option_type_,
            const flatbuffers::Offset< void > & builtin_ops_ )
```

Set the Builtin Options object.

Parameters

*builtin_option_type_*	builtin option type
*builtin_ops_*	flatbuffer offset of builtin ops

Definition at line 82 of file flatbuffer_opnode.cpp.

8.181.3.19 setLayerNode()

```
void nntrainer::FlatBufferOpNode::setLayerNode (
            const LayerNode & layer )
```

Set the Layer Node object.

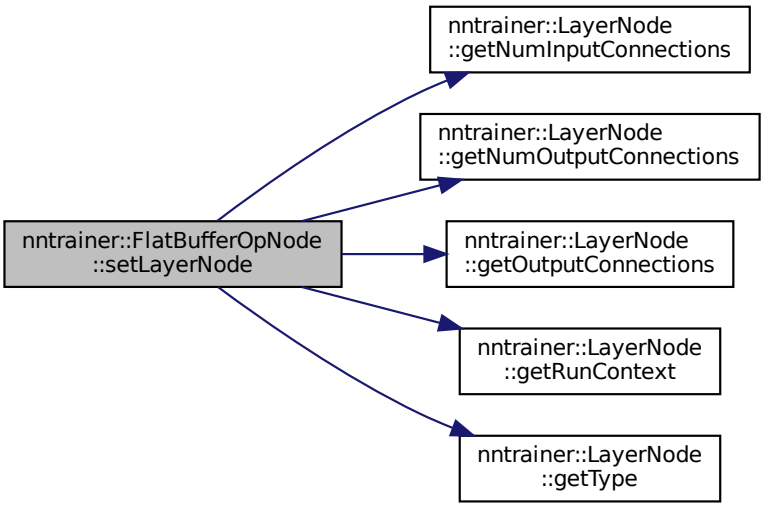
Parameters

layer	layer node

Todo Now support only mse, cross

Definition at line 28 of file flatbuffer_opnode.cpp.

Here is the call graph for this function:



8.181.3.20 setOpType()

```
void nntrainer::FlatBufferOpNode::setOpType (
            nntr::BuiltinOperator op_type_ )  [inline]
```

Set the Op Type object.

Parameters

*op_↩ type_*	

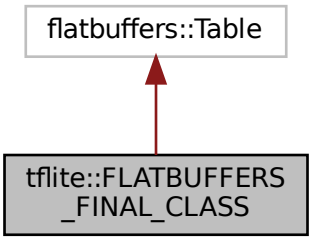
Definition at line 54 of file flatbuffer_opnode.h.

The documentation for this class was generated from the following files:

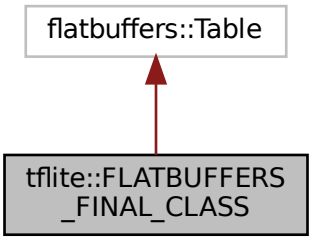
- compiler/flatbuffer_opnode.h
- compiler/flatbuffer_opnode.cpp

8.182 tflite::FLATBUFFERS_FINAL_CLASS Struct Reference

Inheritance diagram for tflite::FLATBUFFERS_FINAL_CLASS:



Collaboration diagram for tflite::FLATBUFFERS_FINAL_CLASS:



Public Types

- typedef CustomQuantizationBuilder **Builder**
- typedef QuantizationParametersBuilder **Builder**
- typedef Int32VectorBuilder **Builder**
- typedef Uint16VectorBuilder **Builder**
- typedef Uint8VectorBuilder **Builder**
- typedef DimensionMetadataBuilder **Builder**
- typedef SparsityParametersBuilder **Builder**
- typedef TensorBuilder **Builder**
- typedef Conv2DOptionsBuilder **Builder**
- typedef Pool2DOptionsBuilder **Builder**
- typedef DepthwiseConv2DOptionsBuilder **Builder**
- typedef ConcatEmbeddingsOptionsBuilder **Builder**
- typedef LSHProjectionOptionsBuilder **Builder**
- typedef SVDFOptionsBuilder **Builder**
- typedef RNNOptionsBuilder **Builder**
- typedef SequenceRNNOptionsBuilder **Builder**
- typedef BidirectionalSequenceRNNOptionsBuilder **Builder**
- typedef FullyConnectedOptionsBuilder **Builder**
- typedef SoftmaxOptionsBuilder **Builder**
- typedef ConcatenationOptionsBuilder **Builder**
- typedef AddOptionsBuilder **Builder**
- typedef MulOptionsBuilder **Builder**
- typedef L2NormOptionsBuilder **Builder**
- typedef LocalResponseNormalizationOptionsBuilder **Builder**
- typedef LSTMOptionsBuilder **Builder**
- typedef UnidirectionalSequenceLSTMOptionsBuilder **Builder**
- typedef BidirectionalSequenceLSTMOptionsBuilder **Builder**
- typedef ResizeBilinearOptionsBuilder **Builder**
- typedef ResizeNearestNeighborOptionsBuilder **Builder**
- typedef CallOptionsBuilder **Builder**
- typedef PadOptionsBuilder **Builder**
- typedef PadV2OptionsBuilder **Builder**
- typedef ReshapeOptionsBuilder **Builder**
- typedef SpaceToBatchNDOptionsBuilder **Builder**
- typedef BatchToSpaceNDOptionsBuilder **Builder**
- typedef SkipGramOptionsBuilder **Builder**
- typedef SpaceToDepthOptionsBuilder **Builder**
- typedef DepthToSpaceOptionsBuilder **Builder**
- typedef SubOptionsBuilder **Builder**
- typedef DivOptionsBuilder **Builder**
- typedef TopKV2OptionsBuilder **Builder**
- typedef EmbeddingLookupSparseOptionsBuilder **Builder**
- typedef GatherOptionsBuilder **Builder**
- typedef TransposeOptionsBuilder **Builder**
- typedef ExpOptionsBuilder **Builder**
- typedef CosOptionsBuilder **Builder**
- typedef ReducerOptionsBuilder **Builder**
- typedef SqueezeOptionsBuilder **Builder**
- typedef SplitOptionsBuilder **Builder**
- typedef SplitVOptionsBuilder **Builder**
- typedef StridedSliceOptionsBuilder **Builder**
- typedef LogSoftmaxOptionsBuilder **Builder**
- typedef CastOptionsBuilder **Builder**

- typedef DequantizeOptionsBuilder **Builder**
- typedef MaximumMinimumOptionsBuilder **Builder**
- typedef TileOptionsBuilder **Builder**
- typedef ArgMaxOptionsBuilder **Builder**
- typedef ArgMinOptionsBuilder **Builder**
- typedef GreaterOptionsBuilder **Builder**
- typedef GreaterEqualOptionsBuilder **Builder**
- typedef LessOptionsBuilder **Builder**
- typedef LessEqualOptionsBuilder **Builder**
- typedef NegOptionsBuilder **Builder**
- typedef SelectOptionsBuilder **Builder**
- typedef SliceOptionsBuilder **Builder**
- typedef TransposeConvOptionsBuilder **Builder**
- typedef ExpandDimsOptionsBuilder **Builder**
- typedef SparseToDenseOptionsBuilder **Builder**
- typedef EqualOptionsBuilder **Builder**
- typedef NotEqualOptionsBuilder **Builder**
- typedef ShapeOptionsBuilder **Builder**
- typedef RankOptionsBuilder **Builder**
- typedef PowOptionsBuilder **Builder**
- typedef FakeQuantOptionsBuilder **Builder**
- typedef PackOptionsBuilder **Builder**
- typedef LogicalOrOptionsBuilder **Builder**
- typedef OneHotOptionsBuilder **Builder**
- typedef AbsOptionsBuilder **Builder**
- typedef HardSwishOptionsBuilder **Builder**
- typedef LogicalAndOptionsBuilder **Builder**
- typedef LogicalNotOptionsBuilder **Builder**
- typedef UnpackOptionsBuilder **Builder**
- typedef FloorDivOptionsBuilder **Builder**
- typedef SquareOptionsBuilder **Builder**
- typedef ZerosLikeOptionsBuilder **Builder**
- typedef FillOptionsBuilder **Builder**
- typedef FloorModOptionsBuilder **Builder**
- typedef RangeOptionsBuilder **Builder**
- typedef LeakyReluOptionsBuilder **Builder**
- typedef SquaredDifferenceOptionsBuilder **Builder**
- typedef MirrorPadOptionsBuilder **Builder**
- typedef UniqueOptionsBuilder **Builder**
- typedef ReverseV2OptionsBuilder **Builder**
- typedef AddNOptionsBuilder **Builder**
- typedef GatherNdOptionsBuilder **Builder**
- typedef WhereOptionsBuilder **Builder**
- typedef ReverseSequenceOptionsBuilder **Builder**
- typedef MatrixDiagOptionsBuilder **Builder**
- typedef QuantizeOptionsBuilder **Builder**
- typedef MatrixSetDiagOptionsBuilder **Builder**
- typedef IfOptionsBuilder **Builder**
- typedef WhileOptionsBuilder **Builder**
- typedef NonMaxSuppressionV4OptionsBuilder **Builder**
- typedef NonMaxSuppressionV5OptionsBuilder **Builder**
- typedef ScatterNdOptionsBuilder **Builder**
- typedef SelectV2OptionsBuilder **Builder**
- typedef DensifyOptionsBuilder **Builder**
- typedef SegmentSumOptionsBuilder **Builder**

- typedef BatchMatMulOptionsBuilder **Builder**
- typedef CumsumOptionsBuilder **Builder**
- typedef OperatorCodeBuilder **Builder**
- typedef OperatorBuilder **Builder**
- typedef SubGraphBuilder **Builder**
- typedef BufferBuilder **Builder**
- typedef MetadataBuilder **Builder**
- typedef TensorMapBuilder **Builder**
- typedef SignatureDefBuilder **Builder**
- typedef ModelBuilder **Builder**

Public Member Functions

- const flatbuffers::Vector< uint8_t > * **custom** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< float > * **min** () const
- const flatbuffers::Vector< float > * **max** () const
- const flatbuffers::Vector< float > * **scale** () const
- const flatbuffers::Vector< int64_t > * **zero_point** () const
- tflite::QuantizationDetails **details_type** () const
- const void * **details** () const
- template<typename T >
  const T * **details_as** () const
- const tflite::CustomQuantization * **details_as_CustomQuantization** () const
- int32_t **quantized_dimension** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< int32_t > * **values** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< uint16_t > * **values** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< uint8_t > * **values** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::DimensionType **format** () const
- int32_t **dense_size** () const
- tflite::SparseIndexVector **array_segments_type** () const
- const void * **array_segments** () const
- template<typename T >
  const T * **array_segments_as** () const
- const tflite::Int32Vector * **array_segments_as_Int32Vector** () const
- const tflite::Uint16Vector * **array_segments_as_Uint16Vector** () const
- const tflite::Uint8Vector * **array_segments_as_Uint8Vector** () const
- tflite::SparseIndexVector **array_indices_type** () const
- const void * **array_indices** () const
- template<typename T >
  const T * **array_indices_as** () const
- const tflite::Int32Vector * **array_indices_as_Int32Vector** () const
- const tflite::Uint16Vector * **array_indices_as_Uint16Vector** () const
- const tflite::Uint8Vector * **array_indices_as_Uint8Vector** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< int32_t > * **traversal_order** () const
- const flatbuffers::Vector< int32_t > * **block_map** () const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::DimensionMetadata > > * **dim_metadata** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< int32_t > * **shape** () const

- tflite::TensorType **type** () const
- uint32_t **buffer** () const
- const flatbuffers::String * **name** () const
- const tflite::QuantizationParameters * **quantization** () const
- bool **is_variable** () const
- const tflite::SparsityParameters * **sparsity** () const
- const flatbuffers::Vector< int32_t > * **shape_signature** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::Padding **padding** () const
- int32_t **stride_w** () const
- int32_t **stride_h** () const
- tflite::ActivationFunctionType **fused_activation_function** () const
- int32_t **dilation_w_factor** () const
- int32_t **dilation_h_factor** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::Padding **padding** () const
- int32_t **stride_w** () const
- int32_t **stride_h** () const
- int32_t **filter_width** () const
- int32_t **filter_height** () const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::Padding **padding** () const
- int32_t **stride_w** () const
- int32_t **stride_h** () const
- int32_t **depth_multiplier** () const
- tflite::ActivationFunctionType **fused_activation_function** () const
- int32_t **dilation_w_factor** () const
- int32_t **dilation_h_factor** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **num_channels** () const
- const flatbuffers::Vector< int32_t > * **num_columns_per_channel** () const
- const flatbuffers::Vector< int32_t > * **embedding_dim_per_channel** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::LSHProjectionType **type** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **rank** () const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **asymmetric_quantize_inputs** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **asymmetric_quantize_inputs** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **time_major** () const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **asymmetric_quantize_inputs** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **time_major** () const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **merge_outputs** () const
- bool **asymmetric_quantize_inputs** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- tflite::FullyConnectedOptionsWeightsFormat **weights_format** () const
- bool **keep_num_dims** () const

- bool **asymmetric_quantize_inputs** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- float **beta** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **axis** () const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **pot_scale_int16** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **radius** () const
- float **bias** () const
- float **alpha** () const
- float **beta** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- float **cell_clip** () const
- float **proj_clip** () const
- tflite::LSTMKernelType **kernel_type** () const
- bool **asymmetric_quantize_inputs** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- float **cell_clip** () const
- float **proj_clip** () const
- bool **time_major** () const
- bool **asymmetric_quantize_inputs** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- float **cell_clip** () const
- float **proj_clip** () const
- bool **merge_outputs** () const
- bool **time_major** () const
- bool **asymmetric_quantize_inputs** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **align_corners** () const
- bool **half_pixel_centers** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **align_corners** () const
- bool **half_pixel_centers** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- uint32_t **subgraph** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< int32_t > * **new_shape** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **ngram_size** () const
- int32_t **max_skip_size** () const
- bool **include_all_ngrams** () const

- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **block_size** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **block_size** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **pot_scale_int16** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::ActivationFunctionType **fused_activation_function** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::CombinerType **combiner** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **axis** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **keep_dims** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< int32_t > * **squeeze_dims** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **num_splits** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **num_splits** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **begin_mask** () const
- int32_t **end_mask** () const
- int32_t **ellipsis_mask** () const
- int32_t **new_axis_mask** () const
- int32_t **shrink_axis_mask** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::TensorType **in_data_type** () const
- tflite::TensorType **out_data_type** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::TensorType **output_type** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::TensorType **output_type** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::Padding **padding** () const
- int32_t **stride_w** () const
- int32_t **stride_h** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const

- bool **validate_indices** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::TensorType **out_type** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- float **min** () const
- float **max** () const
- int32_t **num_bits** () const
- bool **narrow_range** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **values_count** () const
- int32_t **axis** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **axis** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **num** () const
- int32_t **axis** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- float **alpha** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::MirrorPadMode **mode** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- tflite::TensorType **idx_out_type** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **seq_dim** () const
- int32_t **batch_dim** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **then_subgraph_index** () const
- int32_t **else_subgraph_index** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int32_t **cond_subgraph_index** () const
- int32_t **body_subgraph_index** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const

- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **adj_x** () const
- bool **adj_y** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- bool **exclusive** () const
- bool **reverse** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- int8_t **deprecated_builtin_code** () const
- const flatbuffers::String * **custom_code** () const
- int32_t **version** () const
- tflite::BuiltinOperator **builtin_code** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- uint32_t **opcode_index** () const
- const flatbuffers::Vector< int32_t > * **inputs** () const
- const flatbuffers::Vector< int32_t > * **outputs** () const
- tflite::BuiltinOptions **builtin_options_type** () const
- const void * **builtin_options** () const
- template<typename T >
  const T * **builtin_options_as** () const
- const tflite::Conv2DOptions * **builtin_options_as_Conv2DOptions** () const
- const tflite::DepthwiseConv2DOptions * **builtin_options_as_DepthwiseConv2DOptions** () const
- const tflite::ConcatEmbeddingsOptions * **builtin_options_as_ConcatEmbeddingsOptions** () const
- const tflite::LSHProjectionOptions * **builtin_options_as_LSHProjectionOptions** () const
- const tflite::Pool2DOptions * **builtin_options_as_Pool2DOptions** () const
- const tflite::SVDFOptions * **builtin_options_as_SVDFOptions** () const
- const tflite::RNNOptions * **builtin_options_as_RNNOptions** () const
- const tflite::FullyConnectedOptions * **builtin_options_as_FullyConnectedOptions** () const
- const tflite::SoftmaxOptions * **builtin_options_as_SoftmaxOptions** () const
- const tflite::ConcatenationOptions * **builtin_options_as_ConcatenationOptions** () const
- const tflite::AddOptions * **builtin_options_as_AddOptions** () const
- const tflite::L2NormOptions * **builtin_options_as_L2NormOptions** () const
- const tflite::LocalResponseNormalizationOptions * **builtin_options_as_LocalResponseNormalization↩ Options** () const
- const tflite::LSTMOptions * **builtin_options_as_LSTMOptions** () const
- const tflite::ResizeBilinearOptions * **builtin_options_as_ResizeBilinearOptions** () const
- const tflite::CallOptions * **builtin_options_as_CallOptions** () const
- const tflite::ReshapeOptions * **builtin_options_as_ReshapeOptions** () const
- const tflite::SkipGramOptions * **builtin_options_as_SkipGramOptions** () const
- const tflite::SpaceToDepthOptions * **builtin_options_as_SpaceToDepthOptions** () const
- const tflite::EmbeddingLookupSparseOptions * **builtin_options_as_EmbeddingLookupSparseOptions** () const
- const tflite::MulOptions * **builtin_options_as_MulOptions** () const
- const tflite::PadOptions * **builtin_options_as_PadOptions** () const
- const tflite::GatherOptions * **builtin_options_as_GatherOptions** () const
- const tflite::BatchToSpaceNDOptions * **builtin_options_as_BatchToSpaceNDOptions** () const
- const tflite::SpaceToBatchNDOptions * **builtin_options_as_SpaceToBatchNDOptions** () const
- const tflite::TransposeOptions * **builtin_options_as_TransposeOptions** () const
- const tflite::ReducerOptions * **builtin_options_as_ReducerOptions** () const
- const tflite::SubOptions * **builtin_options_as_SubOptions** () const
- const tflite::DivOptions * **builtin_options_as_DivOptions** () const

- const tflite::SqueezeOptions * **builtin_options_as_SqueezeOptions** () const
- const tflite::SequenceRNNOptions * **builtin_options_as_SequenceRNNOptions** () const
- const tflite::StridedSliceOptions * **builtin_options_as_StridedSliceOptions** () const
- const tflite::ExpOptions * **builtin_options_as_ExpOptions** () const
- const tflite::TopKV2Options * **builtin_options_as_TopKV2Options** () const
- const tflite::SplitOptions * **builtin_options_as_SplitOptions** () const
- const tflite::LogSoftmaxOptions * **builtin_options_as_LogSoftmaxOptions** () const
- const tflite::CastOptions * **builtin_options_as_CastOptions** () const
- const tflite::DequantizeOptions * **builtin_options_as_DequantizeOptions** () const
- const tflite::MaximumMinimumOptions * **builtin_options_as_MaximumMinimumOptions** () const
- const tflite::ArgMaxOptions * **builtin_options_as_ArgMaxOptions** () const
- const tflite::LessOptions * **builtin_options_as_LessOptions** () const
- const tflite::NegOptions * **builtin_options_as_NegOptions** () const
- const tflite::PadV2Options * **builtin_options_as_PadV2Options** () const
- const tflite::GreaterOptions * **builtin_options_as_GreaterOptions** () const
- const tflite::GreaterEqualOptions * **builtin_options_as_GreaterEqualOptions** () const
- const tflite::LessEqualOptions * **builtin_options_as_LessEqualOptions** () const
- const tflite::SelectOptions * **builtin_options_as_SelectOptions** () const
- const tflite::SliceOptions * **builtin_options_as_SliceOptions** () const
- const tflite::TransposeConvOptions * **builtin_options_as_TransposeConvOptions** () const
- const tflite::SparseToDenseOptions * **builtin_options_as_SparseToDenseOptions** () const
- const tflite::TileOptions * **builtin_options_as_TileOptions** () const
- const tflite::ExpandDimsOptions * **builtin_options_as_ExpandDimsOptions** () const
- const tflite::EqualOptions * **builtin_options_as_EqualOptions** () const
- const tflite::NotEqualOptions * **builtin_options_as_NotEqualOptions** () const
- const tflite::ShapeOptions * **builtin_options_as_ShapeOptions** () const
- const tflite::PowOptions * **builtin_options_as_PowOptions** () const
- const tflite::ArgMinOptions * **builtin_options_as_ArgMinOptions** () const
- const tflite::FakeQuantOptions * **builtin_options_as_FakeQuantOptions** () const
- const tflite::PackOptions * **builtin_options_as_PackOptions** () const
- const tflite::LogicalOrOptions * **builtin_options_as_LogicalOrOptions** () const
- const tflite::OneHotOptions * **builtin_options_as_OneHotOptions** () const
- const tflite::LogicalAndOptions * **builtin_options_as_LogicalAndOptions** () const
- const tflite::LogicalNotOptions * **builtin_options_as_LogicalNotOptions** () const
- const tflite::UnpackOptions * **builtin_options_as_UnpackOptions** () const
- const tflite::FloorDivOptions * **builtin_options_as_FloorDivOptions** () const
- const tflite::SquareOptions * **builtin_options_as_SquareOptions** () const
- const tflite::ZerosLikeOptions * **builtin_options_as_ZerosLikeOptions** () const
- const tflite::FillOptions * **builtin_options_as_FillOptions** () const
- const tflite::BidirectionalSequenceLSTMOptions * **builtin_options_as_BidirectionalSequenceLSTM↩ Options** () const
- const tflite::BidirectionalSequenceRNNOptions * **builtin_options_as_BidirectionalSequenceRNN↩ Options** () const
- const tflite::UnidirectionalSequenceLSTMOptions * **builtin_options_as_UnidirectionalSequenceLSTM↩ Options** () const
- const tflite::FloorModOptions * **builtin_options_as_FloorModOptions** () const
- const tflite::RangeOptions * **builtin_options_as_RangeOptions** () const
- const tflite::ResizeNearestNeighborOptions * **builtin_options_as_ResizeNearestNeighborOptions** () const
- const tflite::LeakyReluOptions * **builtin_options_as_LeakyReluOptions** () const
- const tflite::SquaredDifferenceOptions * **builtin_options_as_SquaredDifferenceOptions** () const
- const tflite::MirrorPadOptions * **builtin_options_as_MirrorPadOptions** () const
- const tflite::AbsOptions * **builtin_options_as_AbsOptions** () const
- const tflite::SplitVOptions * **builtin_options_as_SplitVOptions** () const
- const tflite::UniqueOptions * **builtin_options_as_UniqueOptions** () const

- const tflite::ReverseV2Options * **builtin_options_as_ReverseV2Options** () const
- const tflite::AddNOptions * **builtin_options_as_AddNOptions** () const
- const tflite::GatherNdOptions * **builtin_options_as_GatherNdOptions** () const
- const tflite::CosOptions * **builtin_options_as_CosOptions** () const
- const tflite::WhereOptions * **builtin_options_as_WhereOptions** () const
- const tflite::RankOptions * **builtin_options_as_RankOptions** () const
- const tflite::ReverseSequenceOptions * **builtin_options_as_ReverseSequenceOptions** () const
- const tflite::MatrixDiagOptions * **builtin_options_as_MatrixDiagOptions** () const
- const tflite::QuantizeOptions * **builtin_options_as_QuantizeOptions** () const
- const tflite::MatrixSetDiagOptions * **builtin_options_as_MatrixSetDiagOptions** () const
- const tflite::HardSwishOptions * **builtin_options_as_HardSwishOptions** () const
- const tflite::IfOptions * **builtin_options_as_IfOptions** () const
- const tflite::WhileOptions * **builtin_options_as_WhileOptions** () const
- const tflite::DepthToSpaceOptions * **builtin_options_as_DepthToSpaceOptions** () const
- const tflite::NonMaxSuppressionV4Options * **builtin_options_as_NonMaxSuppressionV4Options** () const
- const tflite::NonMaxSuppressionV5Options * **builtin_options_as_NonMaxSuppressionV5Options** () const
- const tflite::ScatterNdOptions * **builtin_options_as_ScatterNdOptions** () const
- const tflite::SelectV2Options * **builtin_options_as_SelectV2Options** () const
- const tflite::DensifyOptions * **builtin_options_as_DensifyOptions** () const
- const tflite::SegmentSumOptions * **builtin_options_as_SegmentSumOptions** () const
- const tflite::BatchMatMulOptions * **builtin_options_as_BatchMatMulOptions** () const
- const tflite::CumsumOptions * **builtin_options_as_CumsumOptions** () const
- const flatbuffers::Vector< uint8_t > * **custom_options** () const
- tflite::CustomOptionsFormat **custom_options_format** () const
- const flatbuffers::Vector< uint8_t > * **mutating_variable_inputs** () const
- const flatbuffers::Vector< int32_t > * **intermediates** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::Tensor > > * **tensors** () const
- const flatbuffers::Vector< int32_t > * **inputs** () const
- const flatbuffers::Vector< int32_t > * **outputs** () const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::Operator > > * **operators** () const
- const flatbuffers::String * **name** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< uint8_t > * **data** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::String * **name** () const
- uint32_t **buffer** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::String * **name** () const
- uint32_t **tensor_index** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::TensorMap > > * **inputs** () const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::TensorMap > > * **outputs** () const
- const flatbuffers::String * **method_name** () const
- const flatbuffers::String * **key** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const
- uint32_t **version** () const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::OperatorCode > > * **operator_codes** () const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::SubGraph > > * **subgraphs** () const
- const flatbuffers::String * **description** () const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::Buffer > > * **buffers** () const
- const flatbuffers::Vector< int32_t > * **metadata_buffer** () const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::Metadata > > * **metadata** () const
- const flatbuffers::Vector< flatbuffers::Offset< tflite::SignatureDef > > * **signature_defs** () const
- bool **Verify** (flatbuffers::Verifier &verifier) const

8.182.1 Detailed Description

Definition at line 1965 of file tf_schema_generated.h.

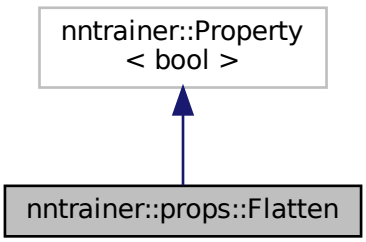
The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

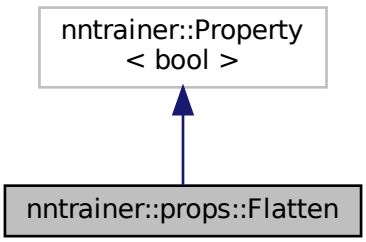
8.183 nntrainer::props::Flatten Class Reference

Flatten property, true if needs flatten layer afterwards.

Inheritance diagram for nntrainer::props::Flatten:



Collaboration diagram for nntrainer::props::Flatten:



Public Types

- using prop_tag = bool_prop_tag

Static Public Attributes

- static constexpr const char * key = "flatten"

8.183.1 Detailed Description

Flatten property, true if needs flatten layer afterwards.

Definition at line 47 of file layer_node.cpp.

8.183.2 Member Typedef Documentation

8.183.2.1 prop_tag

```
using nntrainer::props::Flatten::prop_tag = bool_prop_tag
```

property type

Definition at line 51 of file layer_node.cpp.

8.183.3 Member Data Documentation

8.183.3.1 key

```
constexpr const char* nntrainer::props::Flatten::key = "flatten"  [static], [constexpr]
```

unique key to access

Definition at line 50 of file layer_node.cpp.

The documentation for this class was generated from the following file:

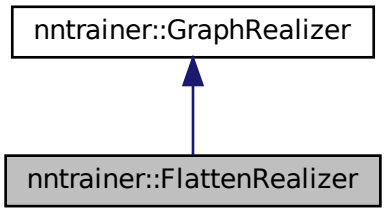
- layers/layer_node.cpp

8.184 nntrainer::FlattenRealizer Class Reference

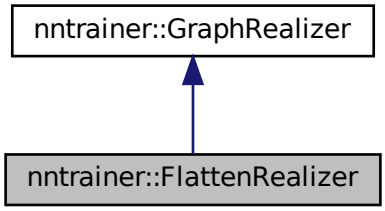
Graph realizer class.

`#include <flatten_realizer.h>`

Inheritance diagram for nntrainer::FlattenRealizer:



Collaboration diagram for nntrainer::FlattenRealizer:



Public Member Functions

- ~FlattenRealizer ()

  Destroy the Graph Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  graph realizer creates a new graph based on the reference

8.184.1 Detailed Description

Graph realizer class.

Definition at line 26 of file flatten_realizer.h.

8.184.2 Constructor & Destructor Documentation

8.184.2.1 ~FlattenRealizer()

```
nntrainer::FlattenRealizer::~FlattenRealizer ( )
```

Destroy the Graph Realizer object.

Definition at line 21 of file flatten_realizer.cpp.

8.184.3 Member Function Documentation

8.184.3.1 realize()

```
GraphRepresentation nntrainer::FlattenRealizer::realize (
            const GraphRepresentation & reference )  [override], [virtual]
```

graph realizer creates a new graph based on the reference

< layer_name

< flatten_layer_name

< temp_layer_name

< layer_name

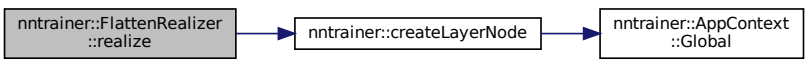
Note

: [node] type=flatten; flatten=true; is awkward but allowed. There is no reason to prohibit this.

Implements nntrainer::GraphRealizer.

Definition at line 24 of file flatten_realizer.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

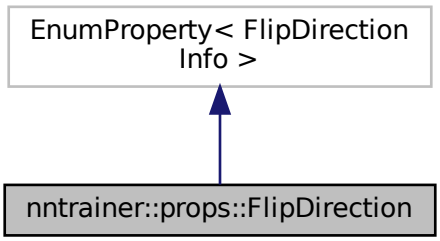
- compiler/flatten_realizer.h
- compiler/flatten_realizer.cpp

8.185 nntrainer::props::FlipDirection Class Reference

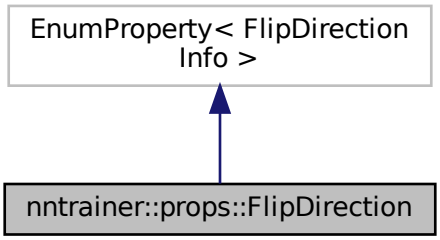
FlipDirection Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::FlipDirection:



Collaboration diagram for nntrainer::props::FlipDirection:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- **FlipDirection** (FlipDirectionInfo::Enum value=FlipDirectionInfo::Enum::horizontal_and_vertical)

Static Public Attributes

- static constexpr const char * **key** = "flip_direction"

8.185.1 Detailed Description

FlipDirection Enumeration Information.

Definition at line 1092 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.186 nntrainer::props::FlipDirectionInfo Struct Reference

Enumeration of flip direction.

`#include <common_properties.h>`

Public Types

- enum **Enum** { **horizontal**, **vertical**, **horizontal_and_vertical** }

Static Public Attributes

- static constexpr std::initializer_list< Enum > **EnumList**
- static constexpr const char * **EnumStr** [ ]

8.186.1 Detailed Description

Enumeration of flip direction.

Definition at line 1079 of file common_properties.h.

8.186.2 Member Data Documentation

8.186.2.1 EnumList

`constexpr std::initializer_list<Enum> nntrainer::props::FlipDirectionInfo::EnumList [static], [constexpr]`

Initial value:

```
= {
    Enum::horizontal, Enum::vertical, Enum::horizontal_and_vertical}
```

Definition at line 1081 of file common_properties.h.

8.186.2.2 EnumStr

```
constexpr const char* nntrainer::props::FlipDirectionInfo::EnumStr[]  [static], [constexpr]
```

Initial value:
```
= {"horizontal", "vertical",
                                              "horizontal_and_vertical"}
```

Definition at line 1084 of file common_properties.h.

The documentation for this struct was generated from the following file:

- layers/common_properties.h

8.187 tflite::FloorDivOptionsBuilder Struct Reference

Public Types

- typedef FloorDivOptions **Table**

Public Member Functions

- **FloorDivOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- FloorDivOptionsBuilder & **operator=** (const FloorDivOptionsBuilder &)
- flatbuffers::Offset< FloorDivOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.187.1 Detailed Description

Definition at line 6120 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.188 tflite::FloorModOptionsBuilder Struct Reference

Public Types

- typedef FloorModOptions **Table**

Public Member Functions

- **FloorModOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- FloorModOptionsBuilder & **operator=** (const FloorModOptionsBuilder &)
- flatbuffers::Offset< FloorModOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.188.1 Detailed Description

Definition at line 6240 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.189 tflite::FullyConnectedOptionsBuilder Struct Reference

Public Types

- typedef FullyConnectedOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_weights_format** (tflite::FullyConnectedOptionsWeightsFormat weights_format)
- void **add_keep_num_dims** (bool keep_num_dims)
- void **add_asymmetric_quantize_inputs** (bool asymmetric_quantize_inputs)
- **FullyConnectedOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- FullyConnectedOptionsBuilder & **operator=** (const FullyConnectedOptionsBuilder &)
- flatbuffers::Offset< FullyConnectedOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.189.1 Detailed Description

Definition at line 3360 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

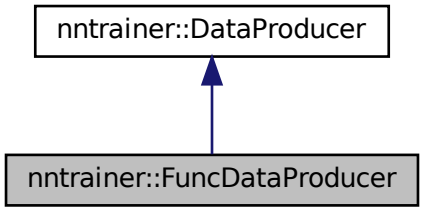
- compiler/tf_schema_generated.h

8.190 nntrainer::FuncDataProducer Class Reference

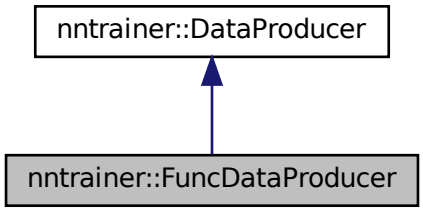
FuncDataProducer which contains a callback and returns back.

`#include <func_data_producer.h>`

Inheritance diagram for nntrainer::FuncDataProducer:



Collaboration diagram for nntrainer::FuncDataProducer:



Public Member Functions

- FuncDataProducer (datagen_cb datagen_cb, void *user_data_)

  Construct a new Func Data Producer object.
- ~FuncDataProducer ()

  Destroy the Func Data Producer object.
- const std::string getType () const override

  Get the producer type.
- void setProperty (const std::vector< std::string > &properties) override
- DataProducer::Generator finalize (const std::vector< TensorDim > &input_dims, const std::vector< TensorDim > &label_dims, void *user_data=nullptr) override
- void exportTo (Exporter &exporter, const ml::train::ExportMethods &method) const override

Static Public Attributes

- static const std::string **type** = "callback"

Additional Inherited Members

8.190.1 Detailed Description

FuncDataProducer which contains a callback and returns back.

Definition at line 34 of file func_data_producer.h.

8.190.2 Constructor & Destructor Documentation

8.190.2.1 FuncDataProducer()

```
nntrainer::FuncDataProducer::FuncDataProducer (
            datagen_cb datagen_cb,
            void * user_data_ )
```

Construct a new Func Data Producer object.

Parameters

datagen_cb	data callback
*user_data_*	user data

Definition at line 22 of file func_data_producer.cpp.

8.190.2.2 ~FuncDataProducer()

```
nntrainer::FuncDataProducer::~FuncDataProducer ( )
```

Destroy the Func Data Producer object.

Definition at line 26 of file func_data_producer.cpp.

8.190.3 Member Function Documentation

8.190.3.1 exportTo()

```
void nntrainer::FuncDataProducer::exportTo (
            Exporter & exporter,
            const ml::train::ExportMethods & method ) const  [override], [virtual]
```

Reimplemented from nntrainer::DataProducer.

Definition at line 73 of file func_data_producer.cpp.

8.190.3.2 finalize()

```
DataProducer::Generator nntrainer::FuncDataProducer::finalize (
            const std::vector< TensorDim > & input_dims,
            const std::vector< TensorDim > & label_dims,
            void * user_data = nullptr )  [override], [virtual]
```

Reimplemented from nntrainer::DataProducer.

Definition at line 39 of file func_data_producer.cpp.

8.190.3.3 getType()

```
const std::string nntrainer::FuncDataProducer::getType ( ) const  [override], [virtual]
```

Get the producer type.

Returns

const std::string type representation

Implements nntrainer::DataProducer.

Definition at line 28 of file func_data_producer.cpp.

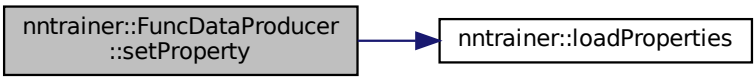
8.190.3.4 setProperty()

```
void nntrainer::FuncDataProducer::setProperty (
            const std::vector< std::string > & properties )  [override], [virtual]
```

Reimplemented from nntrainer::DataProducer.

Definition at line 32 of file func_data_producer.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- dataset/func_data_producer.h
- dataset/func_data_producer.cpp

8.191 tflite::GatherNdOptionsBuilder Struct Reference

Public Types

- typedef GatherNdOptions **Table**

Public Member Functions

- **GatherNdOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- GatherNdOptionsBuilder & **operator=** (const GatherNdOptionsBuilder &)
- flatbuffers::Offset< GatherNdOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.191.1 Detailed Description

Definition at line 6516 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.192 tflite::GatherOptionsBuilder Struct Reference

Public Types

- typedef GatherOptions **Table**

Public Member Functions

- void **add_axis** (int32_t axis)
- **GatherOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- GatherOptionsBuilder & **operator=** (const GatherOptionsBuilder &)
- flatbuffers::Offset< GatherOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.192.1 Detailed Description

Definition at line 4605 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

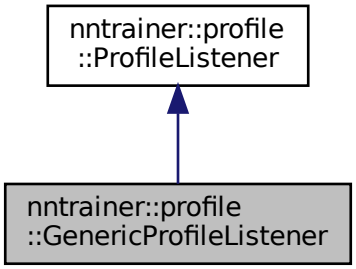
- compiler/tf_schema_generated.h

8.193 nntrainer::profile::GenericProfileListener Class Reference

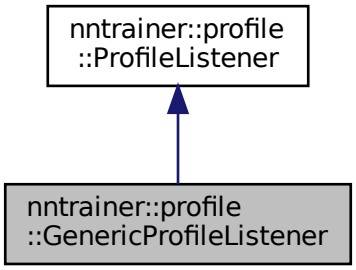
Generic Profiler Listener.

`#include <profiler.h>`

Inheritance diagram for nntrainer::profile::GenericProfileListener:



Collaboration diagram for nntrainer::profile::GenericProfileListener:



Public Member Functions

- GenericProfileListener (int warmups_=0)

  Construct a new GenericProfile Listener object.
- virtual ~GenericProfileListener ()=default

  Destroy the Generic Profile Listener object.
- virtual void notify (PROFILE_EVENT event, const std::shared_ptr< ProfileEventData > data) override

  A callback function to be called from a profiler.
- virtual void reset (const int time_item, const std::string &str) override
- virtual const std::chrono::microseconds result (const int event) override

  get the latest result of a event
- virtual void report (std::ostream &out) const override

8.193.1 Detailed Description

Generic Profiler Listener.

Definition at line 202 of file profiler.h.

8.193.2 Constructor & Destructor Documentation

8.193.2.1 GenericProfileListener()

```
nntrainer::profile::GenericProfileListener::GenericProfileListener (
            int warmups_ = 0 )  [inline], [explicit]
```

Construct a new GenericProfile Listener object.

Parameters

*warmups_*	ignore first *warmups_* records when making time report

Definition at line 209 of file profiler.h.

8.193.2.2 ~GenericProfileListener()

```
virtual nntrainer::profile::GenericProfileListener::~GenericProfileListener ( )  [virtual],
[default]
```

Destroy the Generic Profile Listener object.

8.193.3 Member Function Documentation

8.193.3.1 notify()

```
void nntrainer::profile::GenericProfileListener::notify (
            PROFILE_EVENT event,
            const std::shared_ptr< ProfileEventData > data )  [override], [virtual]
```

A callback function to be called from a profiler.

Parameters

event	event type
data	event data

Implements nntrainer::profile::ProfileListener.

Definition at line 74 of file profiler.cpp.

8.193.3.2 report()

```
void nntrainer::profile::GenericProfileListener::report (
            std::ostream & out ) const  [override], [virtual]
```

creating header

calculate metrics while skipping warmups

creating header

Implements nntrainer::profile::ProfileListener.

Definition at line 118 of file profiler.cpp.

8.193.3.3 reset()

```
void nntrainer::profile::GenericProfileListener::reset (
            const int time_item,
            const std::string & str )  [override], [virtual]
```

Implements nntrainer::profile::ProfileListener.

Definition at line 96 of file profiler.cpp.

8.193.3.4 result()

```
const std::chrono::microseconds nntrainer::profile::GenericProfileListener::result (
            const int event )  [override], [virtual]
```

get the latest result of a event

Parameters

time_item	time item to query the result

Returns

const std::chrono::microseconds

Implements nntrainer::profile::ProfileListener.

Definition at line 105 of file profiler.cpp.

The documentation for this class was generated from the following files:

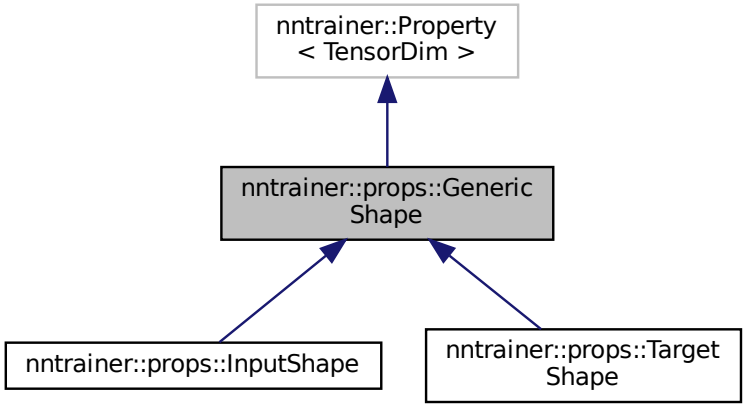
- utils/profiler.h
- utils/profiler.cpp

8.194 nntrainer::props::GenericShape Class Reference

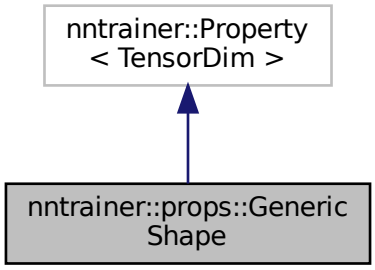
generic shape property which saves a single tensor shape (practically, std::array<GenericShape> is used)

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::GenericShape:



Collaboration diagram for nntrainer::props::GenericShape:



Public Types

- using prop_tag = dimension_prop_tag

Public Member Functions

- void set (const TensorDim &value) override

  Input shape setter.

Static Public Attributes

- static constexpr const char * key

8.194.1 Detailed Description

generic shape property which saves a single tensor shape (practically, std::array<GenericShape> is used)

Note

batch dimension is ignored with this dimension. Setting of batch must be done with the model.

Definition at line 1131 of file common_properties.h.

8.194.2 Member Typedef Documentation

8.194.2.1 prop_tag

```
using nntrainer::props::GenericShape::prop_tag = dimension_prop_tag
```

property type

Definition at line 1136 of file common_properties.h.

8.194.3 Member Function Documentation

8.194.3.1 set()

```
void nntrainer::props::GenericShape::set (
            const TensorDim & value )  [override]
```

Input shape setter.

Parameters

value	value to set

Definition at line 327 of file common_properties.cpp.

8.194.4 Member Data Documentation

8.194.4.1 key

```
constexpr const char* nntrainer::props::GenericShape::key  [static], [constexpr]
```

Initial value:

```
=
    "generic_shape"
```

unique key to access

Definition at line 1134 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.195 nntrainer::GraphCompiler Class Reference

Pure virtual class for the Graph Compiler.

`#include <compiler.h>`

Public Member Functions

- virtual std::shared_ptr< ExecutableGraph > compile (std::shared_ptr< const GraphRepresentation > representation)=0

  serialize graph to a file stream
- virtual std::shared_ptr< GraphRepresentation > decompile (std::shared_ptr< ExecutableGraph > executable)=0

  deserialize graph from a file stream

8.195.1 Detailed Description

Pure virtual class for the Graph Compiler.

Definition at line 51 of file compiler.h.

8.195.2 Member Function Documentation

8.195.2.1 compile()

```
virtual std::shared_ptr<ExecutableGraph> nntrainer::GraphCompiler::compile (
            std::shared_ptr< const GraphRepresentation > representation )  [pure virtual]
```

serialize graph to a file stream

Todo consider adding delegates argument here when implementing it for real.

Parameters

representation	graph representation
file	ifstream to serialize graph

8.195.2.2 decompile()

```
virtual std::shared_ptr<GraphRepresentation> nntrainer::GraphCompiler::decompile (
            std::shared_ptr< ExecutableGraph > executable ) [pure virtual]
```

deserialize graph from a file stream

Parameters

executable	executable graph

Returns

GraphRepresentation graph representation

The documentation for this class was generated from the following file:

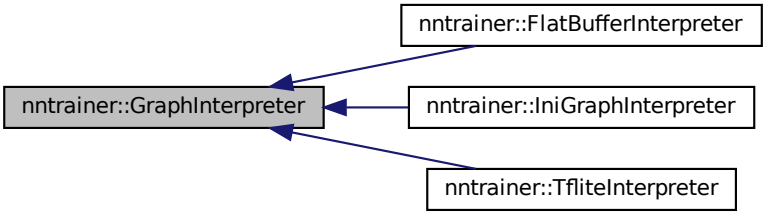
- compiler/compiler.h

8.196 nntrainer::GraphInterpreter Class Reference

Pure virtual class for the Graph Interpreter.

```
#include <interpreter.h>
```

Inheritance diagram for nntrainer::GraphInterpreter:



Public Member Functions

- virtual void serialize (const GraphRepresentation &representation, const std::string &out)=0
  serialize graph to a stream
- virtual void serialize_v1 (const NetworkGraph &representation, const std::string &out)
  serialize graph to a stream
- virtual std::shared_ptr< NetworkGraph > deserialize_v1 (const std::string &in)
  graph representation
- virtual GraphRepresentation deserialize (const std::string &in)=0
  deserialize graph from a stream

8.196.1 Detailed Description

Pure virtual class for the Graph Interpreter.

Definition at line 54 of file interpreter.h.

8.196.2 Member Function Documentation

8.196.2.1 deserialize()

```
virtual GraphRepresentation nntrainer::GraphInterpreter::deserialize (
            const std::string & in )  [pure virtual]
```

deserialize graph from a stream

Parameters

in	input file name

Returns

GraphRepresentation graph representation

Implemented in nntrainer::IniGraphInterpreter, nntrainer::FlatBufferInterpreter, and nntrainer::TfliteInterpreter.

8.196.2.2 deserialize_v1()

```
virtual std::shared_ptr<NetworkGraph> nntrainer::GraphInterpreter::deserialize_v1 (
            const std::string & in )  [inline], [virtual]
```

graph representation

Parameters

in	input file name

Returns

std::shared_ptr<NetworkGraph>

Definition at line 81 of file interpreter.h.

8.196.2.3 serialize()

```
virtual void nntrainer::GraphInterpreter::serialize (
            const GraphRepresentation & representation,
            const std::string & out )  [pure virtual]
```

serialize graph to a stream

Parameters

representation	graph representation
out	output file name

Implemented in nntrainer::IniGraphInterpreter, nntrainer::FlatBufferInterpreter, and nntrainer::TfliteInterpreter.

8.196.2.4 serialize_v1()

```
virtual void nntrainer::GraphInterpreter::serialize_v1 (
            const NetworkGraph & representation,
            const std::string & out )  [inline], [virtual]
```

serialize graph to a stream

Parameters

representation	graph representation
out	output file name

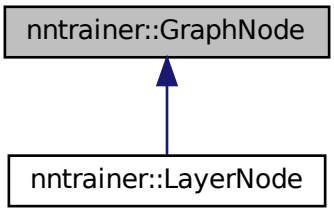
Definition at line 72 of file interpreter.h.

The documentation for this class was generated from the following file:

- compiler/interpreter.h

8.197 nntrainer::GraphNode Class Reference

Inheritance diagram for nntrainer::GraphNode:



Public Types

- typedef std::tuple< unsigned int, unsigned int, unsigned int, unsigned int > ExecutionOrder

  Provides the time/order at which the node will be executed.

Public Member Functions

- virtual ~GraphNode ()=default

  Destructor of Layer Class.
- virtual const std::string getName () const noexcept=0

  Get the Name of the underlying object.
- virtual void setName (const std::string &name)=0

  Set the Name of the underlying object.
- virtual const std::string getType () const =0

  Get the Type of the underlying object.
- virtual bool getTrainable () const =0

  Get the trainable parameter.
- virtual const std::vector< std::string > getInputConnections () const =0

  Get the input connections for this node.
- virtual const std::vector< std::string > getOutputConnections () const =0

  Get the output connections for this node.
- virtual ExecutionOrder getExecutionOrder () const =0

  get the execution order/location of this node
- virtual void setExecutionOrder (ExecutionOrder exec_order_)=0

  set the execution order/location of this node

8.197.1 Detailed Description

Definition at line 27 of file graph_node.h.

8.197.2 Member Typedef Documentation

8.197.2.1 ExecutionOrder

```
typedef std::tuple<unsigned int, unsigned int, unsigned int, unsigned int> nntrainer::GraphNode::ExecutionOrd
```

Provides the time/order at which the node will be executed.

This time will be finalized once the graph has been calculated. Each element indicates the orders with which the below operations for each node are executed:

1. Forwarding
2. calcGradient
3. calcDerivative
4. ApplyGradient One constraint is that they must be sorted in ascending order. This ensures that the operations are executed in the order of their listing.

Definition at line 43 of file graph_node.h.

8.197.3 Member Function Documentation

8.197.3.1 getExecutionOrder()

```
virtual ExecutionOrder nntrainer::GraphNode::getExecutionOrder ( ) const  [pure virtual]
```

get the execution order/location of this node

Return values

the	execution order/location of this node

The two values represents the value for forward and backward respectively

Implemented in nntrainer::LayerNode.

8.197.3.2 getInputConnections()

```
virtual const std::vector<std::string> nntrainer::GraphNode::getInputConnections ( ) const
[pure virtual]
```

Get the input connections for this node.

Returns

list of name of the nodes which form input connections

Implemented in nntrainer::LayerNode.

8.197.3.3 getName()

```
virtual const std::string nntrainer::GraphNode::getName ( ) const  [pure virtual], [noexcept]
```

Get the Name of the underlying object.

Returns

std::string Name of the underlying object

Note

name of each node in the graph must be unique

Implemented in nntrainer::LayerNode.

8.197.3.4 getOutputConnections()

```
virtual const std::vector<std::string> nntrainer::GraphNode::getOutputConnections ( ) const
[pure virtual]
```

Get the output connections for this node.

Returns

list of name of the nodes which form output connections

Implemented in nntrainer::LayerNode.

8.197.3.5 getTrainable()

```
virtual bool nntrainer::GraphNode::getTrainable ( ) const  [pure virtual]
```

Get the trainable parameter.

Returns

bool true / false

Implemented in nntrainer::LayerNode.

8.197.3.6 getType()

```
virtual const std::string nntrainer::GraphNode::getType ( ) const  [pure virtual]
```

Get the Type of the underlying object.

Returns

const std::string type representation

Implemented in nntrainer::LayerNode.

8.197.3.7 setExecutionOrder()

```
virtual void nntrainer::GraphNode::setExecutionOrder (
            ExecutionOrder exec_order_ )  [pure virtual]
```

set the execution order/location of this node

Parameters

exec_order	the execution order/location of this node

The two values represents the value for forward and backward respectively

Implemented in nntrainer::LayerNode.

8.197.3.8 setName()

```
virtual void nntrainer::GraphNode::setName (
            const std::string & name )  [pure virtual]
```

Set the Name of the underlying object.

Parameters

in	*std::string*	Name for the underlying object

Note

name of each node in the graph must be unique, and caller must ensure that

Implemented in nntrainer::LayerNode.

The documentation for this class was generated from the following file:

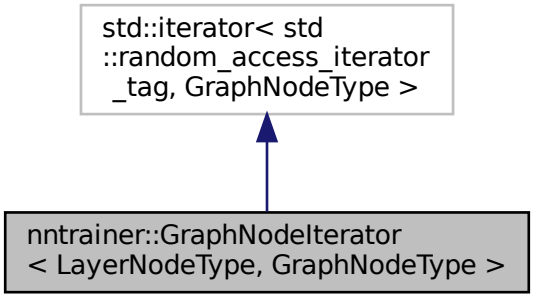
- graph/graph_node.h

8.198 nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType > Class Template Reference

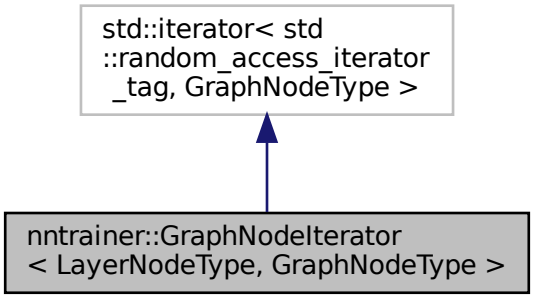
Iterator for GraphNode which return const std::shared_ptr<LayerNodeType> object upon realize.

`#include <graph_node.h>`

Inheritance diagram for nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >:



Collaboration diagram for nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >:



Public Types

- typedef std::random_access_iterator_tag **iterator_category**
- typedef std::ptrdiff_t **difference_type**

Public Member Functions

- GraphNodeIterator (GraphNodeType *x)
  Construct a new Graph Node Iterator object.
- value_type operator* () const
  reference operator
- value_type operator-> () const
  pointer operator
- GraphNodeIterator & operator++ ()
  override for ++ operator
- GraphNodeIterator operator++ (int)
  override for operator++
- GraphNodeIterator & operator-- ()
  override for – operator
- GraphNodeIterator operator-- (int)
  override for operator--
- GraphNodeIterator operator- (const difference_type offset) const
  override for subtract operator
- difference_type operator- (const GraphNodeIterator &other) const
  override for subtract operator
- GraphNodeIterator & operator-= (const difference_type offset)
  override for subtract and return result operator
- GraphNodeIterator operator+ (const difference_type offset) const
  override for add operator
- GraphNodeIterator & operator+= (const difference_type offset)
  override for add and return result operator

Public Attributes

- const typedef std::shared_ptr< LayerNodeType > value_type
  iterator_traits types definition
- const typedef std::shared_ptr< LayerNodeType > * **pointer**
- const typedef std::shared_ptr< LayerNodeType > & **reference**

Friends

- bool operator== (GraphNodeIterator const &lhs, GraphNodeIterator const &rhs)
  == comparison operator override
- bool operator!= (GraphNodeIterator const &lhs, GraphNodeIterator const &rhs)
  != comparison operator override

8.198.1 Detailed Description

template<typename LayerNodeType, typename GraphNodeType>
class nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >

Iterator for GraphNode which return const std::shared_ptr<LayerNodeType> object upon realize.

Note

This does not include the complete list of required functions. Add them as per need.

GraphNodeType is to enable for both GraphNode and const GraphNode

Definition at line 124 of file graph_node.h.

8.198.2 Constructor & Destructor Documentation

8.198.2.1 GraphNodeIterator()

```
template<typename LayerNodeType , typename GraphNodeType >
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::GraphNodeIterator (
            GraphNodeType * x )  [inline]
```

Construct a new Graph Node Iterator object.

Parameters

x	underlying object of GraphNode

Definition at line 149 of file graph_node.h.

8.198.3 Member Function Documentation

8.198.3.1 operator∗()

```
template<typename LayerNodeType , typename GraphNodeType >
value_type nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator* ( ) const
[inline]
```

reference operator

Returns

value_type

Note

this is different from standard iterator

Definition at line 157 of file graph_node.h.

8.198.3.2 operator+()

```
template<typename LayerNodeType , typename GraphNodeType >
GraphNodeIterator nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator+ (
            const difference_type offset ) const  [inline]
```

override for add operator

Parameters

offset	offset to add

Returns

GraphNodeIterator

Definition at line 276 of file graph_node.h.

8.198.3.3 operator++() [1/2]

```
template<typename LayerNodeType , typename GraphNodeType >
GraphNodeIterator& nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator++ (
)  [inline]
```

override for ++ operator

Returns

GraphNodeIterator&

Definition at line 202 of file graph_node.h.

8.198.3.4 operator++() [2/2]

```
template<typename LayerNodeType , typename GraphNodeType >
GraphNodeIterator nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator++ (
            int  )  [inline]
```

override for operator++

Returns

GraphNodeIterator

Definition at line 212 of file graph_node.h.

8.198.3.5 operator+=()

```
template<typename LayerNodeType , typename GraphNodeType >
GraphNodeIterator& nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator+= (
            const difference_type offset )  [inline]
```

override for add and return result operator

Parameters

offset	offset to add

Returns

GraphNodeIterator&

Definition at line 286 of file graph_node.h.

8.198.3.6 operator-() [1/2]

```
template<typename LayerNodeType , typename GraphNodeType >
GraphNodeIterator nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator- (
            const difference_type offset ) const  [inline]
```

override for subtract operator

Parameters

offset	offset to subtract

Returns

GraphNodeIterator

Definition at line 245 of file graph_node.h.

8.198.3.7 operator-() [2/2]

```
template<typename LayerNodeType , typename GraphNodeType >
difference_type nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator- (
            const GraphNodeIterator< LayerNodeType, GraphNodeType > & other ) const  [inline]
```

override for subtract operator

Parameters

other	iterator to subtract

Returns

difference_type

Definition at line 255 of file graph_node.h.

8.198.3.8 operator--() [1/2]

```
template<typename LayerNodeType , typename GraphNodeType >
GraphNodeIterator& nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator-- (
) [inline]
```

override for – operator

Returns

GraphNodeIterator&

Definition at line 223 of file graph_node.h.

8.198.3.9 operator--() [2/2]

```
template<typename LayerNodeType , typename GraphNodeType >
GraphNodeIterator nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator-- (
            int ) [inline]
```

override for operator--

Returns

GraphNodeIterator

Definition at line 233 of file graph_node.h.

8.198.3.10 operator-=()

```
template<typename LayerNodeType , typename GraphNodeType >
GraphNodeIterator& nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator-= (
            const difference_type offset ) [inline]
```

override for subtract and return result operator

Parameters

offset	offset to subtract

Returns

GraphNodeIterator&

Definition at line 265 of file graph_node.h.

8.198.3.11 operator->()

```
template<typename LayerNodeType , typename GraphNodeType >
value_type nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >::operator-> ( ) const
[inline]
```

pointer operator

Returns

value_type

Note

this is different from standard iterator

Definition at line 167 of file graph_node.h.

8.198.4 Friends And Related Function Documentation

8.198.4.1 operator"!=

```
template<typename LayerNodeType , typename GraphNodeType >
bool operator!= (
            GraphNodeIterator< LayerNodeType, GraphNodeType > const & lhs,
            GraphNodeIterator< LayerNodeType, GraphNodeType > const & rhs )  [friend]
```

!= comparison operator override

Parameters

lhs	iterator lhs
rhs	iterator rhs

Return values

true	if mismatch
false	if match

Definition at line 192 of file graph_node.h.

8.198.4.2 operator==

```
template<typename LayerNodeType , typename GraphNodeType >
bool operator== (
            GraphNodeIterator< LayerNodeType, GraphNodeType > const & lhs,
            GraphNodeIterator< LayerNodeType, GraphNodeType > const & rhs )  [friend]
```

== comparison operator override

Parameters

lhs	iterator lhs
rhs	iterator rhs

Return values

true	if match
false	if mismatch

Definition at line 179 of file graph_node.h.

8.198.5 Member Data Documentation

8.198.5.1 value_type

```
template<typename LayerNodeType , typename GraphNodeType >
const typedef std::shared_ptr<LayerNodeType> nntrainer::GraphNodeIterator< LayerNodeType,
GraphNodeType >::value_type
```

iterator_traits types definition

underlying object of GraphNode

Note

these are not requried to be explicitly defined now, but maintains forward compatibility for c++17 and later value_type, pointer and reference are different from standard iterator

Definition at line 138 of file graph_node.h.

The documentation for this class was generated from the following file:

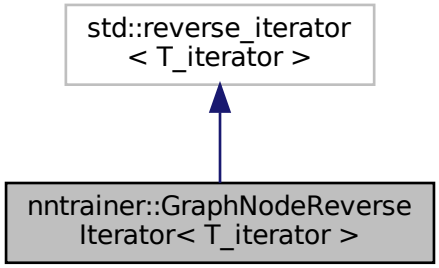
- graph/graph_node.h

8.199 nntrainer::GraphNodeReverseIterator< T_iterator > Class Template Reference

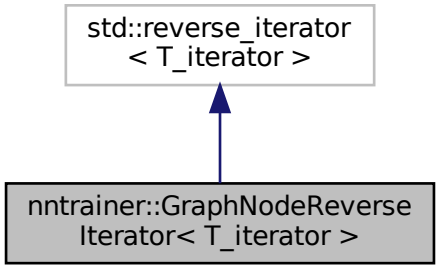
Reverse Iterator for GraphNode which return LayerNode object upon realize.

`#include <graph_node.h>`

Inheritance diagram for nntrainer::GraphNodeReverseIterator< T_iterator >:



Collaboration diagram for nntrainer::GraphNodeReverseIterator< T_iterator >:



Public Member Functions

- GraphNodeReverseIterator (T_iterator iter)

  Construct a new Graph Node Reverse Iterator object.
- T_iterator::value_type operator* () const

  reference operator
- T_iterator::value_type operator-> () const

  pointer operator

8.199.1 Detailed Description

template<typename T_iterator>
class nntrainer::GraphNodeReverseIterator< T_iterator >

Reverse Iterator for GraphNode which return LayerNode object upon realize.

Note

This just extends GraphNodeIterator and is limited by its functionality.

Definition at line 300 of file graph_node.h.

8.199.2 Constructor & Destructor Documentation

8.199.2.1 GraphNodeReverseIterator()

```
template<typename T_iterator >
nntrainer::GraphNodeReverseIterator< T_iterator >::GraphNodeReverseIterator (
            T_iterator iter )  [inline], [explicit]
```

Construct a new Graph Node Reverse Iterator object.

Parameters

iter	Iterator

Definition at line 307 of file graph_node.h.

8.199.3 Member Function Documentation

8.199.3.1 operator*()

```
template<typename T_iterator >
T_iterator::value_type nntrainer::GraphNodeReverseIterator< T_iterator >::operator* ( ) const
[inline]
```

reference operator

Returns

T_iterator::value_type

Note

this is different from standard iterator

Definition at line 316 of file graph_node.h.

8.199.3.2 operator->()

```
template<typename T_iterator >
T_iterator::value_type nntrainer::GraphNodeReverseIterator< T_iterator >::operator-> ( )
const  [inline]
```

pointer operator

Returns

T_iterator::value_type

Note

this is different from standard iterator

Definition at line 327 of file graph_node.h.

The documentation for this class was generated from the following file:

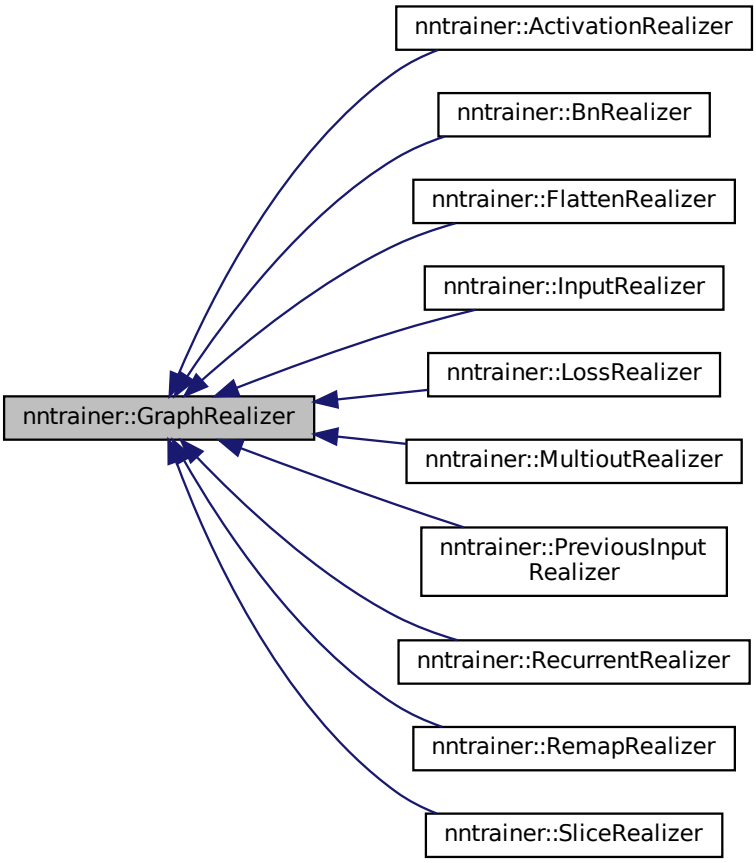
- graph/graph_node.h

8.200 nntrainer::GraphRealizer Class Reference

Graph realizer class.

```
#include <realizer.h>
```

Inheritance diagram for nntrainer::GraphRealizer:



Public Member Functions

- virtual ~GraphRealizer ()

  Destroy the Graph Realizer object.
- virtual GraphRepresentation realize (const GraphRepresentation &reference)=0

  graph realizer creates a new graph based on the reference

8.200.1 Detailed Description

Graph realizer class.

Definition at line 27 of file realizer.h.

8.200.2 Constructor & Destructor Documentation

8.200.2.1 ~GraphRealizer()

```
virtual nntrainer::GraphRealizer::~GraphRealizer ( )  [inline], [virtual]
```

Destroy the Graph Realizer object.

Definition at line 33 of file realizer.h.

8.200.3 Member Function Documentation

8.200.3.1 realize()

```
virtual GraphRepresentation nntrainer::GraphRealizer::realize (
            const GraphRepresentation & reference )  [pure virtual]
```

graph realizer creates a new graph based on the reference

Todo consider void GraphRepresentation &

Implemented in nntrainer::RecurrentRealizer, nntrainer::RemapRealizer, nntrainer::InputRealizer, nntrainer::BnRealizer, nntrainer::SliceRealizer, nntrainer::PreviousInputRealizer, nntrainer::LossRealizer, nntrainer::MultioutRealizer, nntrainer::ActivationRealizer, and nntrainer::FlattenRealizer.

The documentation for this class was generated from the following file:

- compiler/realizer.h

8.201 tflite::GreaterEqualOptionsBuilder Struct Reference

Public Types

- typedef GreaterEqualOptions **Table**

Public Member Functions

- **GreaterEqualOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- GreaterEqualOptionsBuilder & **operator=** (const GreaterEqualOptionsBuilder &)
- flatbuffers::Offset< GreaterEqualOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.201.1 Detailed Description

Definition at line 5276 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.202 tflite::GreaterOptionsBuilder Struct Reference

Public Types

- typedef GreaterOptions **Table**

Public Member Functions

- **GreaterOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- GreaterOptionsBuilder & **operator=** (const GreaterOptionsBuilder &)
- flatbuffers::Offset< GreaterOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.202.1 Detailed Description

Definition at line 5246 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.203 tflite::HardSwishOptionsBuilder Struct Reference

Public Types

- typedef HardSwishOptions **Table**

Public Member Functions

- **HardSwishOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- HardSwishOptionsBuilder & **operator=** (const HardSwishOptionsBuilder &)
- flatbuffers::Offset< HardSwishOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.203.1 Detailed Description

Definition at line 5978 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.204 std::hash< nntrainer::Connection > Struct Reference

hash specialization for connection

`#include <connection.h>`

Public Member Functions

- std::size_t operator() (const nntrainer::Connection &c) const
  hash operator

8.204.1 Detailed Description

hash specialization for connection

Definition at line 128 of file connection.h.

8.204.2 Member Function Documentation

8.204.2.1 operator()()

```
std::size_t std::hash< nntrainer::Connection >::operator() (
            const nntrainer::Connection & c ) const  [inline]
```

hash operator

Parameters

c	connection to hash

Returns

std::size_t hash

Definition at line 135 of file connection.h.

Here is the call graph for this function:

The documentation for this struct was generated from the following file:

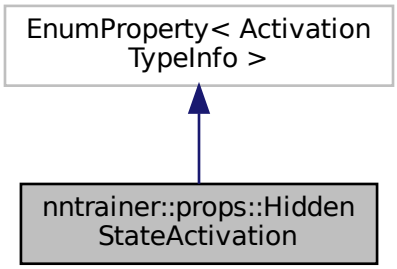
- graph/connection.h

8.205 nntrainer::props::HiddenStateActivation Class Reference

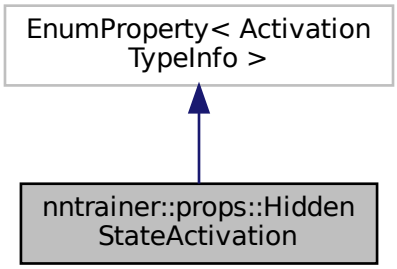
HiddenStateActivation Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::HiddenStateActivation:



Collaboration diagram for nntrainer::props::HiddenStateActivation:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- HiddenStateActivation (ActivationTypeInfo::Enum value=ActivationTypeInfo::Enum::ACT_NONE)
  Construct a new HiddenStateActivation object with default value ActivationTypeInfo::Enum::ACT_NONE.

Static Public Attributes

- static constexpr const char * **key** = "hidden_state_activation"

8.205.1 Detailed Description

HiddenStateActivation Enumeration Information.

Definition at line 865 of file common_properties.h.

8.205.2 Constructor & Destructor Documentation

8.205.2.1 HiddenStateActivation()

```
nntrainer::props::HiddenStateActivation::HiddenStateActivation (
            ActivationTypeInfo::Enum value = ActivationTypeInfo::Enum::ACT_NONE )
```

Construct a new HiddenStateActivation object with default value ActivationTypeInfo::Enum::ACT_NONE.

Definition at line 289 of file common_properties.cpp.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.206 tflite::IfOptionsBuilder Struct Reference

Public Types

- typedef IfOptions **Table**

Public Member Functions

- void **add_then_subgraph_index** (int32_t then_subgraph_index)
- void **add_else_subgraph_index** (int32_t else_subgraph_index)
- **IfOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- IfOptionsBuilder & **operator=** (const IfOptionsBuilder &)
- flatbuffers::Offset< IfOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.206.1 Detailed Description

Definition at line 6730 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

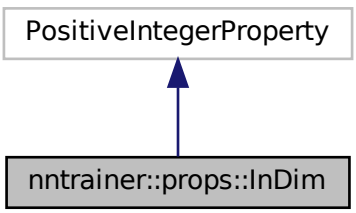
- compiler/tf_schema_generated.h

8.207 nntrainer::props::InDim Class Reference

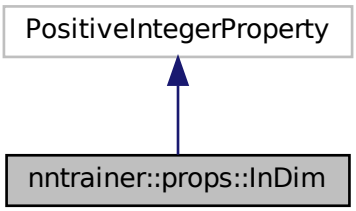
InDim property, in dim is the size of vocabulary in the text data.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::InDim:



Collaboration diagram for nntrainer::props::InDim:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key = "in_dim"

8.207.1 Detailed Description

InDim property, in dim is the size of vocabulary in the text data.

Definition at line 471 of file common_properties.h.

8.207.2 Member Typedef Documentation

8.207.2.1 prop_tag

```
using nntrainer::props::InDim::prop_tag = uint_prop_tag
```

property type

Definition at line 474 of file common_properties.h.

8.207.3 Member Data Documentation

8.207.3.1 key

```
constexpr const char* nntrainer::props::InDim::key = "in_dim"  [static], [constexpr]
```

unique key to access

Definition at line 473 of file common_properties.h.

The documentation for this class was generated from the following file:

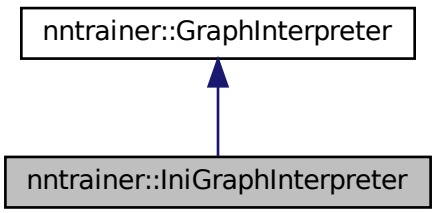
- layers/common_properties.h

8.208 nntrainer::IniGraphInterpreter Class Reference

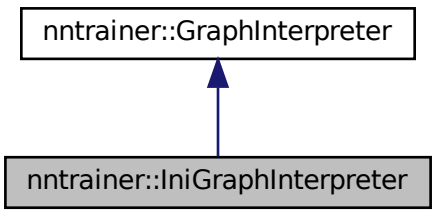
ini graph interpreter class

```
#include <ini_interpreter.h>
```

Inheritance diagram for nntrainer::IniGraphInterpreter:



Collaboration diagram for nntrainer::IniGraphInterpreter:



Public Member Functions

- IniGraphInterpreter (const AppContext &app_context_=AppContext::Global(), std::function< const std::string(const std::string &)> pathResolver_=[](const std::string &path) { return path;})

  Construct a new Ini Graph Interpreter object.
- virtual ~IniGraphInterpreter ()

  Destroy the Ini Graph Interpreter object.
- void serialize (const GraphRepresentation &representation, const std::string &out) override
- GraphRepresentation deserialize (const std::string &in) override

  deserialize graph from a stream

8.208.1 Detailed Description

ini graph interpreter class

Definition at line 30 of file ini_interpreter.h.

8.208.2 Constructor & Destructor Documentation

8.208.2.1 IniGraphInterpreter()

```
nntrainer::IniGraphInterpreter::IniGraphInterpreter (
            const AppContext & app_context_ = AppContext::Global(),
            std::function< const std::string(const std::string &)> pathResolver_ = [](const std::string &path) { return path; } )
```

Construct a new Ini Graph Interpreter object.

Parameters

*pathResolver_*	path resolver function to be used

Definition at line 49 of file ini_interpreter.cpp.

8.208.2.2 ~IniGraphInterpreter()

```
nntrainer::IniGraphInterpreter::~IniGraphInterpreter ( )  [virtual]
```

Destroy the Ini Graph Interpreter object.

Definition at line 55 of file ini_interpreter.cpp.

8.208.3 Member Function Documentation

8.208.3.1 deserialize()

```
GraphRepresentation nntrainer::IniGraphInterpreter::deserialize (
            const std::string & in )  [override], [virtual]
```

deserialize graph from a stream

Parameters

in	input file name

Returns

GraphRepresentation graph representation

Get number of sections in the file

dedicated sections so skip

Parse all the layers defined as sections in order

If this section is a backbone, load backbone section from this

Note

The order of backbones in the ini file defines the order on the backbones in the model graph

Todo : this will be changed to a general way to add a graph

Todo if graph Model Type is of recurrent_wrapper, parse model and realize before return

clean up and rethrow

Implements nntrainer::GraphInterpreter.

Definition at line 285 of file ini_interpreter.cpp.

8.208.3.2 serialize()

```
void nntrainer::IniGraphInterpreter::serialize (
            const GraphRepresentation & representation,
            const std::string & out )  [override], [virtual]
```

Implements nntrainer::GraphInterpreter.

Definition at line 267 of file ini_interpreter.cpp.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- compiler/ini_interpreter.h
- compiler/ini_interpreter.cpp

8.209 nntrainer::IniSection Class Reference

IniSection class that maps to one ini section.

```
#include <ini_wrapper.h>
```

Public Member Functions

- IniSection (const std::string &name)

  Construct a new Ini Section object.
- IniSection (const std::string &section_name, const std::string &entry_str)

  Construct a new Ini Section object.
- IniSection (IniSection &from, const std::string &section_name, const std::string &entry_str)

  Copy Construct a new Ini Section object.
- IniSection (IniSection &from, const std::string &entry_str)

  Construct a new Ini Section object.
- IniSection ()

  Default constructor for the Ini Section object.
- ~IniSection ()=default

  Default destructor for the Ini Section object.
- IniSection & operator+= (const IniSection &rhs)

  +=operator from IniSection
- IniSection operator+ (const IniSection &rhs) const

  - *operator from IniSection*
- IniSection & operator+= (const std::string &s)

  += operator from string
- IniSection operator+ (const std::string &s)
  - *operator from string*
- bool operator== (const IniSection &rhs) const

  equal operator between ini section
- bool operator!= (const IniSection &rhs) const

  not equal operator between ini section
- void print (std::ostream &out) const

  print out a section
- std::string getName () const

  Get the Name object.
- void setEntry (const std::string &key, const std::string &value)

  Set the Entry object by key and value.

Static Public Member Functions

- template<typename Exportable >
  static IniSection FromExportable (const std::string &section_name, const Exportable &exportable)

  Construct a new Ini Section object from which implements object::exportTo.

Friends

- std::ostream & operator<< (std::ostream &os, const IniSection &section)

  <<operator of a section

8.209.1 Detailed Description

IniSection class that maps to one ini section.

Todo consider API style setEntry function

Definition at line 32 of file ini_wrapper.h.

8.209.2 Constructor & Destructor Documentation

8.209.2.1 IniSection() [1/5]

```
nntrainer::IniSection::IniSection (
            const std::string & name )
```

Construct a new Ini Section object.

Parameters

name	section name

Definition at line 23 of file ini_wrapper.cpp.

8.209.2.2 IniSection() [2/5]

```
nntrainer::IniSection::IniSection (
            const std::string & section_name,
            const std::string & entry_str )
```

Construct a new Ini Section object.

Parameters

section_name	section name
entry_str	entry representing string (separated by |)

Definition at line 25 of file ini_wrapper.cpp.

Here is the call graph for this function:

8.209.2.3 IniSection() [3/5]

```
nntrainer::IniSection::IniSection (
            IniSection & from,
            const std::string & section_name,
            const std::string & entry_str )
```

Copy Construct a new Ini Section object.

Note

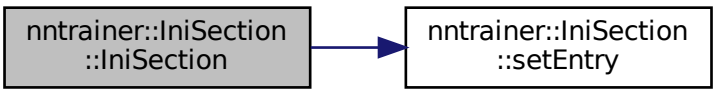
this function copies entry from *from* and overwrite entry and section_name

Parameters

from	Ini Section to copy from
section_name	section name to override, if empty, section name is not updated
entry_str	entry string to override the given section name

Definition at line 31 of file ini_wrapper.cpp.

Here is the call graph for this function:



8.209.2.4 IniSection() [4/5]

```
nntrainer::IniSection::IniSection (
            IniSection & from,
            const std::string & entry_str )  [inline]
```

Construct a new Ini Section object.

Parameters

from	Ini Section to copy from
entry_str	entry string to override the given section name

Definition at line 69 of file ini_wrapper.h.

8.209.2.5 IniSection() [5/5]

```
nntrainer::IniSection::IniSection ( )  [inline]
```

Default constructor for the Ini Section object.

Definition at line 76 of file ini_wrapper.h.

Here is the caller graph for this function:

8.209.2.6 ~IniSection()

```
nntrainer::IniSection::~IniSection ( )  [default]
```

Default destructor for the Ini Section object.

8.209.3 Member Function Documentation

8.209.3.1 FromExportable()

```
template<typename Exportable >
static IniSection nntrainer::IniSection::FromExportable (
            const std::string & section_name,
            const Exportable & exportable )  [inline], [static]
```

Construct a new Ini Section object from which implements object::exportTo.

Template Parameters

Exportable	object with member object::exportTo

Parameters

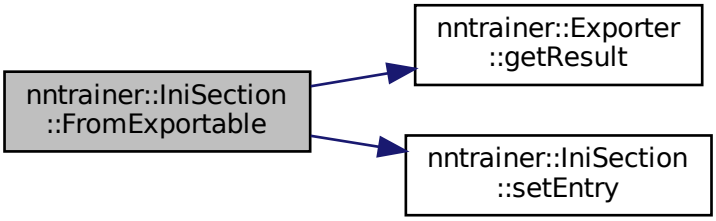
section_name	section name
exportable	exportable object

Returns

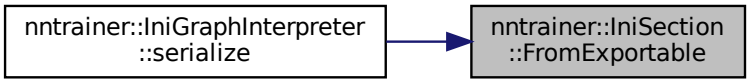
IniSection created section

Definition at line 88 of file ini_wrapper.h.

Here is the call graph for this function:



Here is the caller graph for this function:



8.209.3.2 getName()

```
std::string nntrainer::IniSection::getName ( ) const  [inline]
```

Get the Name object.

Returns

std::string section name

Definition at line 184 of file ini_wrapper.h.

8.209.3.3 operator"!=()

```
bool nntrainer::IniSection::operator!= (
            const IniSection & rhs ) const  [inline]
```

not equal operator between ini section

Parameters

rhs	ini section to compare

Return values

true	two inisections are not equal
false	two inisections are equal

Definition at line 170 of file ini_wrapper.h.

Here is the call graph for this function:

8.209.3.4 operator+() [1/2]

```
IniSection nntrainer::IniSection::operator+ (
            const IniSection & rhs ) const  [inline]
```

- operator from IniSection

Parameters

rhs	operand to add

Returns

IniSection new Inisection

Definition at line 129 of file ini_wrapper.h.

Here is the call graph for this function:

8.209.3.5 operator+() [2/2]

```
IniSection nntrainer::IniSection::operator+ (
            const std::string & s )  [inline]
```

- operator from string

Parameters

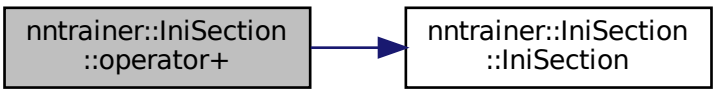
s	string representation to add

Returns

IniSection Newly created section

Definition at line 150 of file ini_wrapper.h.

Here is the call graph for this function:



8.209.3.6 operator+=() [1/2]

```
IniSection& nntrainer::IniSection::operator+= (
            const IniSection & rhs )  [inline]
```

+=operator from IniSection

Parameters

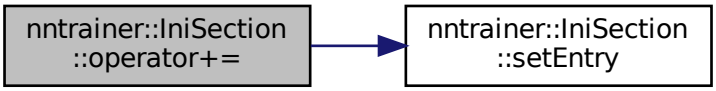
rhs	operand to add

Returns

IniSection& this

Definition at line 118 of file ini_wrapper.h.

Here is the call graph for this function:



8.209.3.7 operator+=() [2/2]

```
IniSection& nntrainer::IniSection::operator+= (
            const std::string & s )  [inline]
```

+= operator from string

Parameters

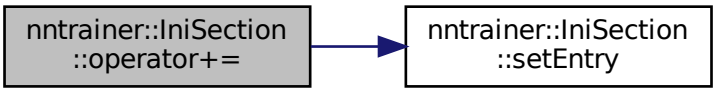
s	string representation to add

Returns

IniSection& this

Definition at line 139 of file ini_wrapper.h.

Here is the call graph for this function:



8.209.3.8 operator==()

```
bool nntrainer::IniSection::operator== (
            const IniSection & rhs ) const  [inline]
```

equal operator between ini section

Parameters

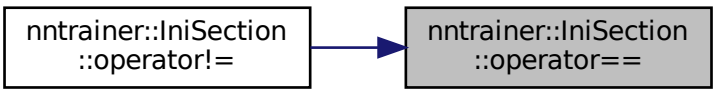
rhs	ini section to compare

Return values

true	two inisections are equal
false	two ini sections are not equal

Definition at line 159 of file ini_wrapper.h.

Here is the caller graph for this function:



8.209.3.9 print()

```
void nntrainer::IniSection::print (
            std::ostream & out ) const
```

print out a section

Parameters

out	ostream to print

Definition at line 42 of file ini_wrapper.cpp.

8.209.3.10 setEntry()

```
void nntrainer::IniSection::setEntry (
            const std::string & key,
            const std::string & value )
```

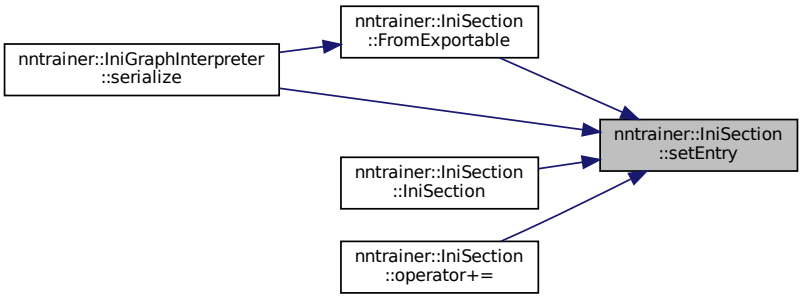
Set the Entry object by key and value.

Parameters

key	key to update
value	value to be added

Definition at line 54 of file ini_wrapper.cpp.

Here is the caller graph for this function:



8.209.4 Friends And Related Function Documentation

8.209.4.1 operator<<

```
std::ostream& operator<< (
             std::ostream & os,
             const IniSection & section )  [friend]
```

<<operator of a section

Parameters

os	ostream
section	section to print

Returns

std::ostream& ostream

Definition at line 223 of file ini_wrapper.h.

The documentation for this class was generated from the following files:

- utils/ini_wrapper.h
- utils/ini_wrapper.cpp

8.210 nntrainer::props::InitializerInfo Struct Reference

Enumeration of tensor initialization type.

`#include <common_properties.h>`

Public Types

- using **Enum** = Tensor::Initializer

Static Public Attributes

- static constexpr std::initializer_list< Enum > **EnumList**
- static constexpr const char * **EnumStr** [ ]

8.210.1 Detailed Description

Enumeration of tensor initialization type.

Definition at line 898 of file common_properties.h.

8.210.2 Member Data Documentation

8.210.2.1 EnumList

```
constexpr std::initializer_list<Enum> nntrainer::props::InitializerInfo::EnumList  [static],
[constexpr]
```

Initial value:

```
= {
    Enum::ZEROS,          Enum::ONES,           Enum::LECUN_NORMAL,
    Enum::LECUN_UNIFORM, Enum::XAVIER_NORMAL, Enum::XAVIER_UNIFORM,
    Enum::HE_NORMAL,      Enum::HE_UNIFORM,     Enum::NONE}
```

Definition at line 900 of file common_properties.h.

8.210.2.2 EnumStr

```
constexpr const char* nntrainer::props::InitializerInfo::EnumStr[]  [static], [constexpr]
```

Initial value:

```
= {
    "zeros",          "ones",           "lecun_normal",
    "lecun_uniform", "xavier_normal", "xavier_uniform",
    "he_normal",      "he_uniform",     "none"}
```

Definition at line 905 of file common_properties.h.

The documentation for this struct was generated from the following file:

- layers/common_properties.h

8.211 nntrainer::InitLayerContext Class Reference

Public Types

- typedef VarGradSpec TensorSpec

  Specification of the tensors.

Public Member Functions

- InitLayerContext (const std::vector< TensorDim > &dim, const std::vector< bool > &req_out_connected, bool in_place_, const std::string &n="", const std::string &prefix_="", const float max_norm=0.0)

  Construct a new Init Layer Context object.
- const std::string & getName () const

  get name by the layer
- unsigned int getNumInputs () const

  Get the number of inputs for the layer.
- unsigned int getNumRequestedOutputs () const

  Get the number of requested outputs for the layer.
- const std::vector< TensorDim > & getInputDimensions () const

  Get the Input Dimensions object.
- void setEffDimFlagInputDimension (unsigned int idx, const std::bitset< TensorDim::MAXDIM > &dim_flag_)

  Set the Dim Flag to retrieve effective dimension.
- void setDynDimFlagInputDimension (unsigned int idx, const std::bitset< TensorDim::MAXDIM > &dim_↩ flag_)

  Set the dynamic Dim Flag to retrieve dynamic dimension (that can change during running)
- void setOutputDimensions (const std::vector< TensorDim > &out_dim)

  Set the Output Dimensions object.
- unsigned int requestWeight (const TensorDim &dim, const Tensor::Initializer init, const WeightRegularizer reg, const float reg_const, const float decay, const std::string &name, bool trainable=true)

  Request a new weight for the layer.
- unsigned int requestWeight (const WeightSpec &spec)

  Request a new weight for the layer.
- unsigned int requestTensor (const TensorDim &dim, const std::string &name, const Tensor::Initializer init=Tensor::Initializer::NONE, bool trainable=false, TensorLifespan lifespan=TensorLifespan::ITERATION_LIFESPAN, bool private_=true)

  Request a new tensor for the layer.
- unsigned int requestTensor (const TensorSpec &spec)

  Request a new tensor for the layer.
- const std::vector< WeightSpec > & getWeightsSpec () const

  Get the current weights spec.
- unsigned int getNumWeights () const

  Get the number of requested weights.
- const std::vector< TensorSpec > & getTensorsSpec () const

  Get the current tensors spec.
- unsigned int getNumTensors () const

  Get the number of requested tensors objects.
- void requestOutputs (std::vector< VarGradSpecV2 > &&out_specs)

  request outputs
- const std::vector< VarGradSpecV2 > & getOutSpecs () const

  Get the Out Specs object.
- bool validate ()

  Validate the context.
- bool executeInPlace () const

  check if the layer is expected to run in-place

Static Public Member Functions

- static VarGradSpecV2 outSpec (const TensorDim &dim, const std::string &name="out", TensorLifespan ls=TensorLifespan::FORWARD_FUNC_LIFESPAN, TensorLifespan grad_ls=TensorLifespan::CALC_GRAD_DERIV_LIFESPAN

  create var grad specification with output default

8.211.1 Detailed Description

Definition at line 38 of file layer_context.h.

8.211.2 Member Typedef Documentation

8.211.2.1 TensorSpec

```
typedef VarGradSpec nntrainer::InitLayerContext::TensorSpec
```

Specification of the tensors.

Definition at line 182 of file layer_context.h.

8.211.3 Constructor & Destructor Documentation

8.211.3.1 InitLayerContext()

```
nntrainer::InitLayerContext::InitLayerContext (
            const std::vector< TensorDim > & dim,
            const std::vector< bool > & req_out_connected,
            bool in_place_,
            const std::string & n = "",
            const std::string & prefix_ = "",
            const float max_norm = 0.0 )
```

Construct a new Init Layer Context object.

Parameters

dim	Input dimensions for the layer
req_out_connected	bool vector to tell if requested output is trainable or not
*in_place_*	true if the context is inplacable
name	name
*prefix_*	prefix
max_norm	max norm

Definition at line 40 of file layer_context.cpp.

8.211.4 Member Function Documentation

8.211.4.1 executeInPlace()

```
bool nntrainer::InitLayerContext::executeInPlace ( ) const  [inline]
```

check if the layer is expected to run in-place

Returns

true if in-place, else false

Definition at line 284 of file layer_context.h.

8.211.4.2 getInputDimensions()

```
const std::vector<TensorDim>& nntrainer::InitLayerContext::getInputDimensions ( ) const  [inline]
```

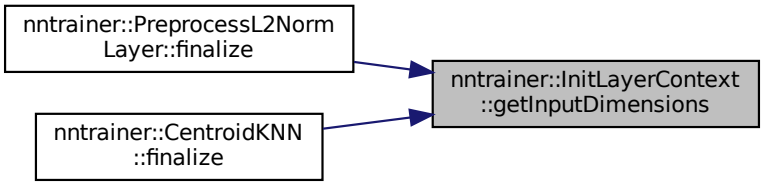
Get the Input Dimensions object.

Returns

const std::vector<TensorDim>& Input dimensions

Definition at line 82 of file layer_context.h.

Here is the caller graph for this function:



8.211.4.3 getName()

```
const std::string& nntrainer::InitLayerContext::getName ( ) const  [inline]
```

get name by the layer

Returns

name of the layer

Definition at line 61 of file layer_context.h.

8.211.4.4 getNumInputs()

```
unsigned int nntrainer::InitLayerContext::getNumInputs ( ) const  [inline]
```

Get the number of inputs for the layer.

Returns

unsigned int number of inputs

Definition at line 68 of file layer_context.h.

8.211.4.5 getNumRequestedOutputs()

```
unsigned int nntrainer::InitLayerContext::getNumRequestedOutputs ( ) const
```

Get the number of requested outputs for the layer.

Returns

unsigned int number of requested outputs

Definition at line 59 of file layer_context.cpp.

8.211.4.6 getNumTensors()

```
unsigned int nntrainer::InitLayerContext::getNumTensors ( ) const  [inline]
```

Get the number of requested tensors objects.

Returns

unsigned int number of requested tensors

Definition at line 224 of file layer_context.h.

8.211.4.7 getNumWeights()

```
unsigned int nntrainer::InitLayerContext::getNumWeights ( ) const  [inline]
```

Get the number of requested weights.

Returns

The current number of requested weights

Definition at line 210 of file layer_context.h.

8.211.4.8 getOutSpecs()

```
const std::vector< VarGradSpecV2 > & nntrainer::InitLayerContext::getOutSpecs ( ) const
```

Get the Out Specs object.

Returns

std::vector<VarGradSpecV2> out specification

Definition at line 118 of file layer_context.cpp.

8.211.4.9 getTensorsSpec()

```
const std::vector<TensorSpec>& nntrainer::InitLayerContext::getTensorsSpec ( ) const  [inline]
```

Get the current tensors spec.

Returns

The current tensors spec

Definition at line 217 of file layer_context.h.

8.211.4.10 getWeightsSpec()

```
const std::vector<WeightSpec>& nntrainer::InitLayerContext::getWeightsSpec ( ) const  [inline]
```

Get the current weights spec.

Returns

The current weights spec

Definition at line 203 of file layer_context.h.

8.211.4.11 outSpec()

```
VarGradSpecV2 nntrainer::InitLayerContext::outSpec (
            const TensorDim & dim,
            const std::string & name = "out",
            TensorLifespan ls = TensorLifespan::FORWARD_FUNC_LIFESPAN,
            TensorLifespan grad_ls = TensorLifespan::CALC_GRAD_DERIV_LIFESPAN )  [static]
```

create var grad specification with output default

Parameters

dim	dimension
name	name
ls	variable lifespan
grad_ls	gradient lifespan

Returns

VarGradSpecV2 var grad specification

Definition at line 76 of file layer_context.cpp.

Here is the caller graph for this function:

8.211.4.12 requestOutputs()

```
void nntrainer::InitLayerContext::requestOutputs (
            std::vector< VarGradSpecV2 > && out_specs )
```

request outputs

Parameters

out_specs	pack of out specification, name will be automatically indexed to prevent name clash

Definition at line 90 of file layer_context.cpp.

Here is the caller graph for this function:

8.211.4.13 requestTensor() [1/2]

```
unsigned int nntrainer::InitLayerContext::requestTensor (
            const TensorDim & dim,
            const std::string & name,
            const Tensor::Initializer init = Tensor::Initializer::NONE,
            bool trainable = false,
            TensorLifespan lifespan = TensorLifespan::ITERATION_LIFESPAN,
            bool private_ = true )  [inline]
```

Request a new tensor for the layer.

Parameters

dim	dimension of the tensor
trainable	if the tensor is trainable (require gradient or not)
name	name of the tensor
lifespan	lifespan of the tensor
*private_*	if custom tensor should not be shared, and only for soleuse

Returns

unsigned int index of the tensor for its getter

Todo Consider providing a guarantee that the returned indices will always start from 0 and will always be incremental.

Definition at line 167 of file layer_context.h.

8.211.4.14 requestTensor() [2/2]

```
unsigned int nntrainer::InitLayerContext::requestTensor (
            const TensorSpec & spec )  [inline]
```

Request a new tensor for the layer.

Parameters

spec	tensor spec

Returns

unsigned int index of the tensor for its getter

Todo Consider providing a guarantee that the returned indices will always start from 0 and will always be incremental.

Definition at line 193 of file layer_context.h.

8.211.4.15 requestWeight() [1/2]

```
unsigned int nntrainer::InitLayerContext::requestWeight (
            const TensorDim & dim,
            const Tensor::Initializer init,
            const WeightRegularizer reg,
            const float reg_const,
            const float decay,
            const std::string & name,
            bool trainable = true )  [inline]
```

Request a new weight for the layer.

Parameters

dim	dimension of the weight
init	initializer for the weight
reg	regularizer for the weight
reg_const	regularization constant for the weight
name	name of the weight
trainable	if the weight is trainable (require gradient or not)

Returns

unsigned int index of the weight for its getter

Todo Consider providing a guarantee that the returned indices will always start from 0 and will always be incremental.

Definition at line 128 of file layer_context.h.

8.211.4.16 requestWeight() [2/2]

```
unsigned int nntrainer::InitLayerContext::requestWeight (
            const WeightSpec & spec )  [inline]
```

Request a new weight for the layer.

Parameters

spec	tensor spec

Returns

unsigned int index of the weight for its getter

Todo Consider providing a guarantee that the returned indices will always start from 0 and will always be incremental.

Definition at line 148 of file layer_context.h.

8.211.4.17 setDynDimFlagInputDimension()

```
void nntrainer::InitLayerContext::setDynDimFlagInputDimension (
            unsigned int idx,
            const std::bitset< TensorDim::MAXDIM > & dim_flag_ )  [inline]
```

Set the dynamic Dim Flag to retrieve dynamic dimension (that can change during running)

Parameters

*dim_↩ flag_*	dimension bit to calculate, rightmost is width

Definition at line 102 of file layer_context.h.

8.211.4.18 setEffDimFlagInputDimension()

```
void nntrainer::InitLayerContext::setEffDimFlagInputDimension (
            unsigned int idx,
            const std::bitset< TensorDim::MAXDIM > & dim_flag_ )  [inline]
```

Set the Dim Flag to retrieve effective dimension.

Parameters

*dim_↩ flag_*	dimension bit to calculate, rightmost is width

Definition at line 90 of file layer_context.h.

8.211.4.19 setOutputDimensions()

```
void nntrainer::InitLayerContext::setOutputDimensions (
            const std::vector< TensorDim > & out_dim )
```

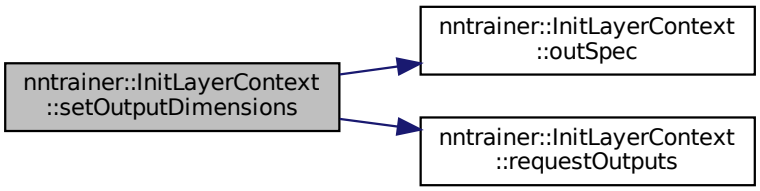
Set the Output Dimensions object.

Parameters

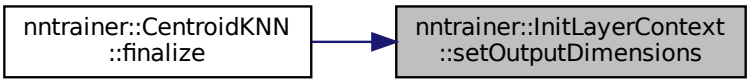
out_dim	the output dimension to set to

Definition at line 63 of file layer_context.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.211.4.20 validate()

```
bool nntrainer::InitLayerContext::validate ( )  [inline]
```

Validate the context.

Returns

true if validated, else false

Note

this must be called before passing a context to a layer for finalize

Definition at line 261 of file layer_context.h.

The documentation for this class was generated from the following files:

- layers/layer_context.h
- layers/layer_context.cpp

8.212 nntrainer::IniWrapper Class Reference

IniWrapper using IniSections.

`#include <ini_wrapper.h>`

Public Types

- using **Sections** = std::vector< IniSection >

Public Member Functions

- IniWrapper ()=default

  Construct a new Ini Test Wrapper object.
- IniWrapper (const std::string &name_, const Sections &sections_={})

  Construct a new Ini Test Wrapper object.
- bool operator== (const IniWrapper &rhs) const

  ini operator== to check if IniWrapper is equal
- bool operator!= (const IniWrapper &rhs) const

  ini operator!= to check if IniWrapper is not equal
- IniWrapper & operator+= (const IniWrapper &ini)

  update sections if section is empty, else update section by section by key
- IniWrapper operator+ (const IniWrapper &rhs) const

  update sections if section is empty, else update section by section by key
- IniWrapper & operator+= (const std::string &s)

  update a single section using operator+=
- IniWrapper & operator+= (const IniSection &section_)

  update a single section using operator+=
- IniWrapper operator+ (const std::string &rhs) const

  update a single section using operator +
- IniWrapper operator+ (const IniSection &section_) const

  update a single section using operator +
- std::string getIniName () const

  Get the Ini Name object.
- std::string getName () const

  Get the Name.
- void save_ini () const

  save ini to a file, (getIniName() is used to save)
- void save_ini (const std::string &ini_name) const

  save ini by ini_name
- void erase_ini () const noexcept

  erase ini

Friends

- std::ostream & operator<< (std::ostream &os, const IniWrapper &ini)

  operator<< to print information to outstream

8.212.1 Detailed Description

IniWrapper using IniSections.

Definition at line 232 of file ini_wrapper.h.

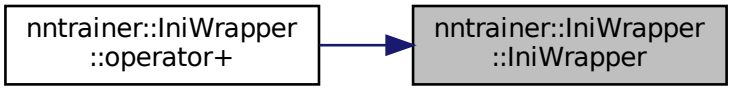
8.212.2 Constructor & Destructor Documentation

8.212.2.1 IniWrapper() [1/2]

```
nntrainer::IniWrapper::IniWrapper ( )  [default]
```

Construct a new Ini Test Wrapper object.

Here is the caller graph for this function:



8.212.2.2 IniWrapper() [2/2]

```
nntrainer::IniWrapper::IniWrapper (
            const std::string & name_,
            const Sections & sections_ = {} )  [inline]
```

Construct a new Ini Test Wrapper object.

Parameters

*name_*	name of the ini without `.ini` extension
*sections_*	sections that should go into ini

Definition at line 248 of file ini_wrapper.h.

8.212.3 Member Function Documentation

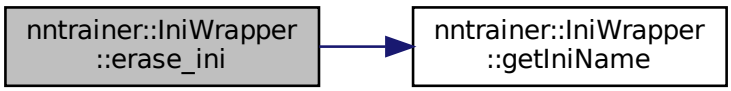
8.212.3.1 erase_ini()

```
void nntrainer::IniWrapper::erase_ini ( ) const  [noexcept]
```

erase ini

Definition at line 129 of file ini_wrapper.cpp.

Here is the call graph for this function:



8.212.3.2 getIniName()

```
std::string nntrainer::IniWrapper::getIniName ( ) const  [inline]
```

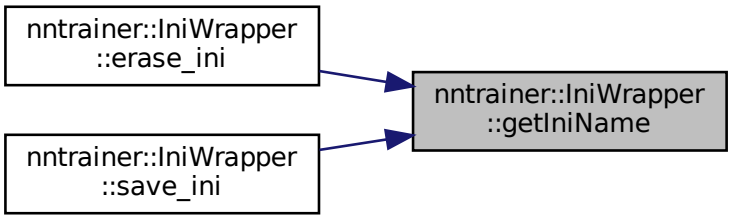
Get the Ini Name object.

Returns

std::string ini name with extension appended

Definition at line 347 of file ini_wrapper.h.

Here is the caller graph for this function:



8.212.3.3 getName()

```
std::string nntrainer::IniWrapper::getName ( ) const  [inline]
```

Get the Name.

Returns

std::string name

Definition at line 354 of file ini_wrapper.h.

8.212.3.4 operator"!=()

```
bool nntrainer::IniWrapper::operator!= (
            const IniWrapper & rhs ) const  [inline]
```

ini operator!= to check if IniWrapper is not equal

Parameters

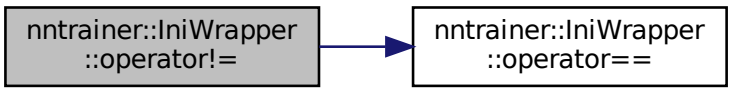
rhs	IniWrapper to compare

Return values

true	if not equal
false	if equal

Definition at line 270 of file ini_wrapper.h.

Here is the call graph for this function:



8.212.3.5 operator+() [1/3]

```
IniWrapper nntrainer::IniWrapper::operator+ (
            const IniSection & section_ ) const  [inline]
```

update a single section using operator +

Parameters

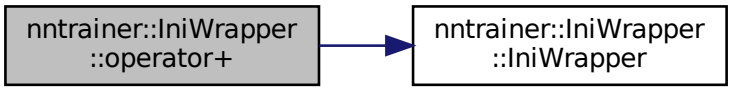
rhs	string representatioin to merge

Returns

IniWrapper ini wrapper

Definition at line 338 of file ini_wrapper.h.

Here is the call graph for this function:



8.212.3.6 operator+() [2/3]

```
IniWrapper nntrainer::IniWrapper::operator+ (
            const IniWrapper & rhs ) const  [inline]
```

update sections if section is empty, else update section by section by key

Parameters

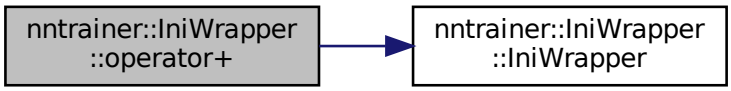
in	*rhs*	IniWrapper

Returns

IniWrapper& a new instance

Definition at line 296 of file ini_wrapper.h.

Here is the call graph for this function:



8.212.3.7 operator+() [3/3]

```
IniWrapper nntrainer::IniWrapper::operator+ (
            const std::string & rhs ) const  [inline]
```

update a single section using operator +

Parameters

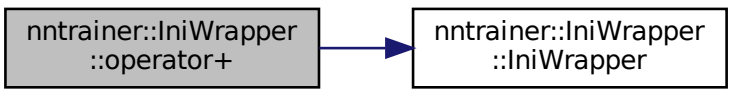
rhs	string representatioin to merge

Returns

IniWrapper ini wrapper

Definition at line 328 of file ini_wrapper.h.

Here is the call graph for this function:



8.212.3.8 operator+=() [1/3]

```
IniWrapper& nntrainer::IniWrapper::operator+= (
            const IniSection & section_ )  [inline]
```

update a single section using operator+=

Parameters

string	format of `sectionkey / propkey=val | propkey=val| ..`

Returns

IniWrapper& ini wrapper

Definition at line 317 of file ini_wrapper.h.

8.212.3.9 operator+=() [2/3]

```
IniWrapper& nntrainer::IniWrapper::operator+= (
            const IniWrapper & ini )  [inline]
```

update sections if section is empty, else update section by section by key

Parameters

in	*ini*	IniWrapper

Returns

IniWrapper& this

Definition at line 279 of file ini_wrapper.h.

8.212.3.10 operator+=() [3/3]

```
IniWrapper& nntrainer::IniWrapper::operator+= (
            const std::string & s )  [inline]
```

update a single section using operator+=

Parameters

string	format of `sectionkey / propkey=val | propkey=val| ..`

Returns

IniWrapper& ini wrapper

Definition at line 306 of file ini_wrapper.h.

8.212.3.11 operator==()

```
bool nntrainer::IniWrapper::operator== (
            const IniWrapper & rhs ) const  [inline]
```

ini operator== to check if IniWrapper is equal

Parameters

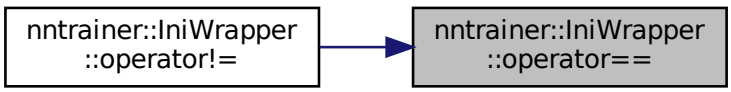
rhs	IniWrapper to compare

Return values

true	true if ini is equal (deeply)
false	false if ini is not equal

Definition at line 259 of file ini_wrapper.h.

Here is the caller graph for this function:



8.212.3.12 save_ini()

```
void nntrainer::IniWrapper::save_ini (
            const std::string & ini_name ) const
```

save ini by ini_name

Parameters

ini_name	ini name to svae

Definition at line 120 of file ini_wrapper.cpp.

8.212.4 Friends And Related Function Documentation

8.212.4.1 operator<<

```
std::ostream& operator<< (
             std::ostream & os,
             const IniWrapper & ini )  [friend]
```

operator<< to print information to outstream

Parameters

os	outstream
ini	ini wrapper

Returns

std::ostream& outstream

Definition at line 381 of file ini_wrapper.h.

The documentation for this class was generated from the following files:

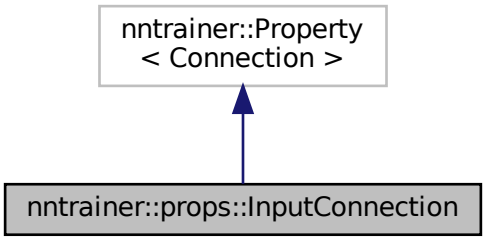
- utils/ini_wrapper.h
- utils/ini_wrapper.cpp

8.213 nntrainer::props::InputConnection Class Reference

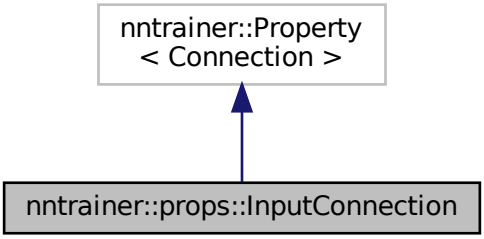
InputSpec property, this defines connection specification of an input.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::InputConnection:



Collaboration diagram for nntrainer::props::InputConnection:



Public Types

- using prop_tag = connection_prop_tag

Public Member Functions

- InputConnection ()

  Construct a new Input Spec object.
- InputConnection (const Connection &value)

  Construct a new Input Spec object.

Static Public Attributes

- static constexpr const char * key

8.213.1 Detailed Description

InputSpec property, this defines connection specification of an input.

Definition at line 183 of file common_properties.h.

8.213.2 Member Typedef Documentation

8.213.2.1 prop_tag

```
using nntrainer::props::InputConnection::prop_tag = connection_prop_tag
```

property type

Definition at line 199 of file common_properties.h.

8.213.3 Constructor & Destructor Documentation

8.213.3.1 InputConnection() [1/2]

```
nntrainer::props::InputConnection::InputConnection ( )
```

Construct a new Input Spec object.

Definition at line 85 of file common_properties.cpp.

8.213.3.2 InputConnection() [2/2]

```
nntrainer::props::InputConnection::InputConnection (
            const Connection & value )
```

Construct a new Input Spec object.

Parameters

value	default value of a input spec

8.213.4 Member Data Documentation

8.213.4.1 key

```
constexpr const char* nntrainer::props::InputConnection::key  [static], [constexpr]
```

Initial value:

```
=
    "input_layers"
```

unique key to access

Definition at line 197 of file common_properties.h.

The documentation for this class was generated from the following files:

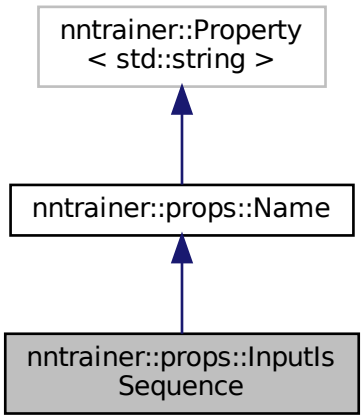
- layers/common_properties.h
- layers/common_properties.cpp

8.214 nntrainer::props::InputIsSequence Class Reference

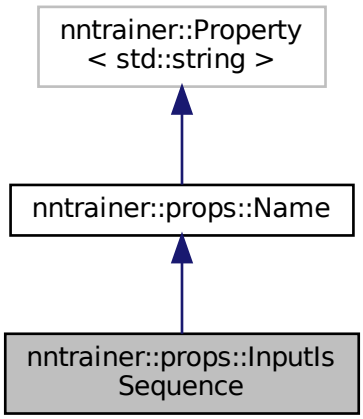
Identifiers to locate an **input** connection which should be sequenced for the connection.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::InputIsSequence:



Collaboration diagram for nntrainer::props::InputIsSequence:



Public Types

- using **prop_tag** = str_prop_tag

Static Public Attributes

- static constexpr const char * **key** = "input_is_sequence"

Additional Inherited Members

8.214.1 Detailed Description

Identifiers to locate an **input** connection which should be sequenced for the connection.

Definition at line 675 of file common_properties.h.

The documentation for this class was generated from the following file:

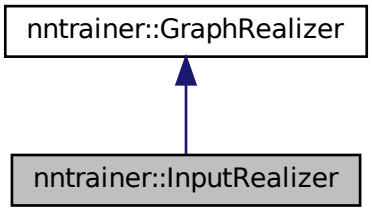
- layers/common_properties.h

8.215 nntrainer::InputRealizer Class Reference

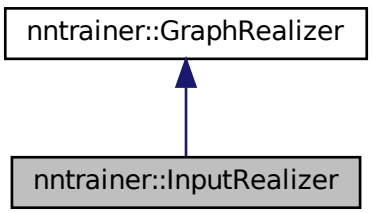
Graph realizer class which remaps input from start -> input layers.

`#include <input_realizer.h>`

Inheritance diagram for nntrainer::InputRealizer:



Collaboration diagram for nntrainer::InputRealizer:



Public Member Functions

- InputRealizer (const std::vector< Connection > &start_conns, const std::vector< Connection > &input_↩ conns)

  Construct a new Input Realizer object.
- ~InputRealizer ()

  Destroy the Graph Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  graph realizer creates a shallow copied graph based on the reference

8.215.1 Detailed Description

Graph realizer class which remaps input from start -> input layers.

Note

This class overwrites input conns to the location of start conns. This requires each start location have the slot for input connection.

When number of input connection == 0 for a start connection of index zero, this pushes input connection to the slot

Definition at line 32 of file input_realizer.h.

8.215.2 Constructor & Destructor Documentation

8.215.2.1 InputRealizer()

```
nntrainer::InputRealizer::InputRealizer (
            const std::vector< Connection > & start_conns,
            const std::vector< Connection > & input_conns )
```

Construct a new Input Realizer object.

Parameters

start_conns	start layers
input_conns	input layers

Definition at line 23 of file input_realizer.cpp.

8.215.2.2 ~InputRealizer()

```
nntrainer::InputRealizer::~InputRealizer ( )
```

Destroy the Graph Realizer object.

Definition at line 31 of file input_realizer.cpp.

8.215.3 Member Function Documentation

8.215.3.1 realize()

```
GraphRepresentation nntrainer::InputRealizer::realize (
            const GraphRepresentation & reference )  [override], [virtual]
```

graph realizer creates a shallow copied graph based on the reference

Note

input realizer resets input_layers of start_layers so that it can be connected to the external network

Exceptions

std::invalid_argument	if graph is ill formed

Implements nntrainer::GraphRealizer.

Definition at line 34 of file input_realizer.cpp.

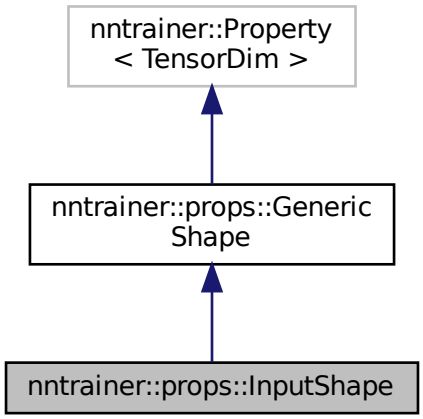
The documentation for this class was generated from the following files:

- compiler/input_realizer.h
- compiler/input_realizer.cpp

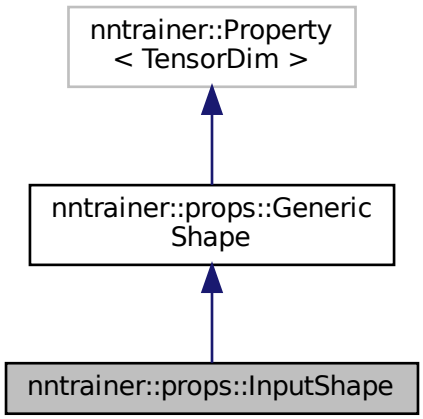
8.216 nntrainer::props::InputShape Class Reference

Input shape property which saves a single tensor shape (practically, std::array<InputShape> is used)

Inheritance diagram for nntrainer::props::InputShape:



Collaboration diagram for nntrainer::props::InputShape:



Public Types

- using prop_tag = dimension_prop_tag

Static Public Attributes

- static constexpr const char * key = "input_shape"

Additional Inherited Members

8.216.1 Detailed Description

Input shape property which saves a single tensor shape (practically, std::array<InputShape> is used)

Definition at line 102 of file layer_node.cpp.

8.216.2 Member Typedef Documentation

8.216.2.1 prop_tag

```
using nntrainer::props::InputShape::prop_tag = dimension_prop_tag
```

property type

Definition at line 106 of file layer_node.cpp.

8.216.3 Member Data Documentation

8.216.3.1 key

```
constexpr const char* nntrainer::props::InputShape::key = "input_shape"  [static], [constexpr]
```

unique key to access

Definition at line 105 of file layer_node.cpp.

The documentation for this class was generated from the following file:

- layers/layer_node.cpp

8.217 tflite::Int32VectorBuilder Struct Reference

Public Types

- typedef Int32Vector **Table**

Public Member Functions

- void **add_values** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> values)
- **Int32VectorBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- Int32VectorBuilder & **operator=** (const Int32VectorBuilder &)
- flatbuffers::Offset< Int32Vector > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.217.1 Detailed Description

Definition at line 2173 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

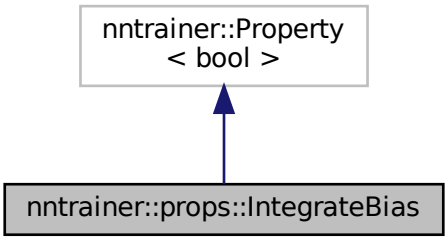
- compiler/tf_schema_generated.h

8.218 nntrainer::props::IntegrateBias Class Reference

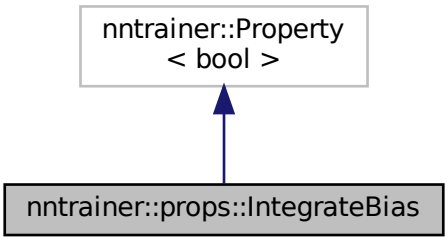
Integrate bias_ih and bias_hh to bias_h to use only 1 bias (Used in rnn variant)

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::IntegrateBias:



Collaboration diagram for nntrainer::props::IntegrateBias:



Public Types

- using **prop_tag** = bool_prop_tag

Public Member Functions

- IntegrateBias (bool val=false)

  Construct a IntegrateBias object.

Static Public Attributes

- static constexpr const char * **key** = "integrate_bias"

8.218.1 Detailed Description

Integrate bias_ih and bias_hh to bias_h to use only 1 bias (Used in rnn variant)

Definition at line 130 of file common_properties.h.

8.218.2 Constructor & Destructor Documentation

8.218.2.1 IntegrateBias()

```
nntrainer::props::IntegrateBias::IntegrateBias (
            bool val = false )  [inline]
```

Construct a IntegrateBias object.

Definition at line 136 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.219 nntrainer::Iteration Class Reference

Iteration class which owns the memory chunk for a single batch.

`#include <data_iteration.h>`

Public Member Functions

- Iteration (const std::vector< ml::train::TensorDim > &input_dims, const std::vector< ml::train::TensorDim > &label_dims)

  Construct a new Iteration object.
- **Iteration** (const Iteration &rhs)=delete
- Iteration & **operator=** (const Iteration &rhs)=delete
- **Iteration** (Iteration &&rhs)=default
- Iteration & **operator=** (Iteration &&rhs)=default
- unsigned int batch ()

  get batch size of iteration
- std::vector< Tensor > & getInputsRef ()

  Get the Input Reference object.
- const std::vector< Tensor > & getInputsRef () const

  Get the Input Reference object.
- std::vector< Tensor > & getLabelsRef ()

  Get the Label Reference object.
- const std::vector< Tensor > & getLabelsRef () const

  Get the Label Reference object.
- std::vector< Sample >::iterator begin ()

  get sample iterator begin()
- std::vector< Sample >::iterator end ()

  get sample iterator end
- std::vector< Sample >::const_iterator begin () const

  get sample iterator begin
- std::vector< Sample >::const_iterator end () const

  get sample iterator end
- void setEndSample (std::vector< Sample >::iterator sample_iterator)

  set end of the sample which will be used to calculate the batch size
- void setEndSample ()

  Set the End Sample to the original end.

8.219.1 Detailed Description

Iteration class which owns the memory chunk for a single batch.

Definition at line 32 of file data_iteration.h.

8.219.2 Constructor & Destructor Documentation

8.219.2.1 Iteration()

```
nntrainer::Iteration::Iteration (
            const std::vector< ml::train::TensorDim > & input_dims,
            const std::vector< ml::train::TensorDim > & label_dims )
```

Construct a new Iteration object.

Note

the batch dimension must be the same for all given dimensions and the there must be at least one input

Parameters

input_dims	input dimension
label_dims	label dimension

Definition at line 76 of file data_iteration.cpp.

8.219.3 Member Function Documentation

8.219.3.1 batch()

```
unsigned int nntrainer::Iteration::batch ( )
```

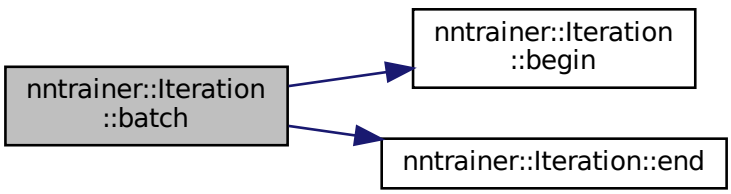
get batch size of iteration

Returns

unsigned int batch size

Definition at line 88 of file data_iteration.cpp.

Here is the call graph for this function:



8.219.3.2 begin() [1/2]

```
std::vector<Sample>::iterator nntrainer::Iteration::begin ( )  [inline]
```

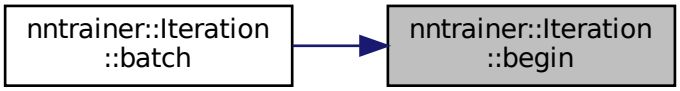
get sample iterator begin()

Returns

std::vector<Sample>::iterator

Definition at line 91 of file data_iteration.h.

Here is the caller graph for this function:



8.219.3.3 begin() [2/2]

```
std::vector<Sample>::const_iterator nntrainer::Iteration::begin ( ) const  [inline]
```

get sample iterator begin

Returns

std::vector<Sample>::const_iterator

Definition at line 105 of file data_iteration.h.

8.219.3.4 end() [1/2]

```
std::vector<Sample>::iterator nntrainer::Iteration::end ( )  [inline]
```

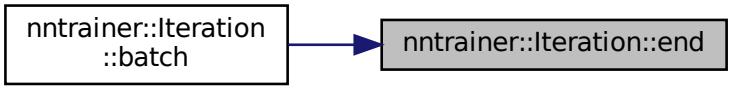
get sample iterator end

Returns

std::vector<Sample>::iterator

Definition at line 98 of file data_iteration.h.

Here is the caller graph for this function:



8.219.3.5 end() [2/2]

```
std::vector<Sample>::const_iterator nntrainer::Iteration::end ( ) const  [inline]
```

get sample iterator end

Returns

std::vector<Sample>::const_iterator

Definition at line 112 of file data_iteration.h.

8.219.3.6 getInputsRef() [1/2]

```
std::vector<Tensor>& nntrainer::Iteration::getInputsRef ( )  [inline]
```

Get the Input Reference object.

Returns

std::vector<Tensor>& input

Definition at line 63 of file data_iteration.h.

8.219.3.7 getInputsRef() [2/2]

```
const std::vector<Tensor>& nntrainer::Iteration::getInputsRef ( ) const  [inline]
```

Get the Input Reference object.

Returns

const std::vector<Tensor>& input

Definition at line 70 of file data_iteration.h.

8.219.3.8 getLabelsRef() [1/2]

```
std::vector<Tensor>& nntrainer::Iteration::getLabelsRef ( )  [inline]
```

Get the Label Reference object.

Returns

std::vector<Tensor>& label

Definition at line 77 of file data_iteration.h.

8.219.3.9 getLabelsRef() [2/2]

```
const std::vector<Tensor>& nntrainer::Iteration::getLabelsRef ( ) const  [inline]
```

Get the Label Reference object.

Returns

const std::vector<Tensor>& label

Definition at line 84 of file data_iteration.h.

8.219.3.10 setEndSample() [1/2]

```
void nntrainer::Iteration::setEndSample ( )
```

Set the End Sample to the original end.

Definition at line 94 of file data_iteration.cpp.

8.219.3.11 setEndSample() [2/2]

```
void nntrainer::Iteration::setEndSample (
            std::vector< Sample >::iterator sample_iterator )
```

set end of the sample which will be used to calculate the batch size

Note

iteration must be non-inclusive

Definition at line 90 of file data_iteration.cpp.

The documentation for this class was generated from the following files:

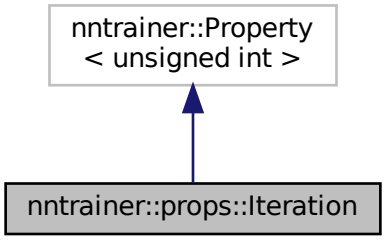
- dataset/data_iteration.h
- dataset/data_iteration.cpp

8.220 nntrainer::props::Iteration Class Reference

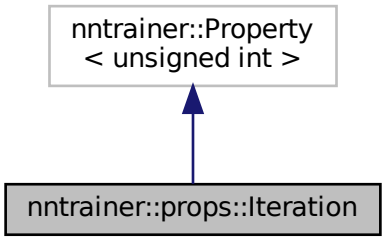
Iteration props.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Iteration:



Collaboration diagram for nntrainer::props::Iteration:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key = "iteration"

8.220.1 Detailed Description

Iteration props.

Definition at line 1295 of file common_properties.h.

8.220.2 Member Typedef Documentation

8.220.2.1 prop_tag

```
using nntrainer::props::Iteration::prop_tag = uint_prop_tag
```

property type

Definition at line 1298 of file common_properties.h.

8.220.3 Member Data Documentation

8.220.3.1 key

```
constexpr const char* nntrainer::props::Iteration::key = "iteration"  [static], [constexpr]
```

unique key to access

Definition at line 1297 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.221 nntrainer::IterationQueue Class Reference

Iteration queue that owns the buffer for input / labels.

`#include <iteration_queue.h>`

Public Member Functions

- IterationQueue (unsigned int num_slots, const std::vector< ml::train::TensorDim > &input_dims, const std::vector< ml::train::TensorDim > &label_dims)

  Construct a new Iteration Queue object.
- ~IterationQueue ()

  Destroy the Iteration Queue object.
- ScopedView< Sample > requestEmptySlot ()

  request empty sample from the queue.
- ScopedView< Iteration > requestFilledSlot ()

  request filled iteration from the queue.
- unsigned int slots ()

  get slot size, slot size is number of batches inside the queue
- unsigned int batch ()

  get size of batch for one iteration
- void notifyEndOfRequestEmpty ()

  notifyEndOfRequest, when the producing by requestEmptySlot has finished.

8.221.1 Detailed Description

Iteration queue that owns the buffer for input / labels.

- requestEmptySlot() will give a ScopedView<sample> Destructing the returned object will notify the iteration that is done filling the sample. Once iteration is done filling, it will internally call IterationQueue::markFilled();
- requestFilledSlot() will give a ScopedView<Iteration> Destructing this will notify the queue that is done used (internally calls IterationQueue::markEmpty())

For an iteration there can be four state.

1. The buffer is empty, waiting to be filled (will be in empty_q)
2. The buffer is being filled sample by sample, waiting to be marked as filled.
3. The buffer is filled, waiting to be served (will be in filled_q)
4. The buffer is being served, waiting to be marked as emptied.

Todo apply this to the databuffer

handle error case: 1. when ScopedView<Sample> has met throw

(a) when ScopedView<Iteration> has met throw

Definition at line 199 of file iteration_queue.h.

8.221.2 Constructor & Destructor Documentation

8.221.2.1 IterationQueue()

```
nntrainer::IterationQueue::IterationQueue (
            unsigned int num_slots,
            const std::vector< ml::train::TensorDim > & input_dims,
            const std::vector< ml::train::TensorDim > & label_dims )
```

Construct a new Iteration Queue object.

Note

input_dimension and label_dimension should include the batch, if IterationQueue::batch() is zero, it means it's invalid

Parameters

num_slots	number of slots this iteration queue will allocate, it should be buffersize/batchsize
input_dims	input dimensions
label_dims	label dimensions

Definition at line 24 of file iteration_queue.cpp.

8.221.2.2 ~IterationQueue()

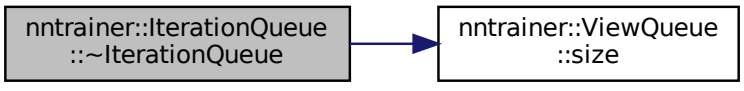
```
nntrainer::IterationQueue::~IterationQueue ( )
```

Destroy the Iteration Queue object.

if an iteration is not included in either empty_q or filled_q, that means it's either being filled or being served. Which means it will be dangerous to destroy *this*, we might want to wait on the destructor if we can assure this can stay no except

Definition at line 41 of file iteration_queue.cpp.

Here is the call graph for this function:



8.221.3 Member Function Documentation

8.221.3.1 batch()

```
unsigned int nntrainer::IterationQueue::batch ( )  [inline]
```

get size of batch for one iteration

Returns

unsigned int size of batch

Definition at line 253 of file iteration_queue.h.

8.221.3.2 notifyEndOfRequestEmpty()

```
void nntrainer::IterationQueue::notifyEndOfRequestEmpty ( )
```

notifyEndOfRequest, when the producing by requestEmptySlot has finished.

Note

It is important that the owner of this class must ensure that there will be no more requestEmptySlot call after this. This means that, in case of multiple workers, the manager of the worker(producer) must know every producer has finished. and call this function other than each worker call this function.

we have to defined ordering of having stop_requested -> push nullptr to filled_q -> stopped so when the case of changing to stopped it has to push nullptr to empty_q, and filled_q to wake them up and stop. this has potential cases that weren't considered. let's change this to a simpler mechanisms to wait on conditional variable.

below is useful information when debugging iteration queue, but there will be too much log if we turn the log on. so leaving it as a comment for now.

Definition at line 129 of file iteration_queue.cpp.

8.221.3.3 requestEmptySlot()

```
ScopedView< Sample > nntrainer::IterationQueue::requestEmptySlot ( )
```

request empty sample from the queue.

Note

User must check if ScopedView actually has a value by calling ScopedView::isEmpty()

Returns

ScopedView<Sample> sample view. ScopedView::isEmpty() == true if there is no more data coming. Destroying the returned object will signal the queue that the sample is filled.

below is useful information when debugging iteration queue, but there will be too much log if we turn the log on. so leaving it as a comment for now.

Definition at line 54 of file iteration_queue.cpp.

8.221.3.4 requestFilledSlot()

```
ScopedView< Iteration > nntrainer::IterationQueue::requestFilledSlot ( )
```

request filled iteration from the queue.

Note

User must check if ScopedView actually has a value by calling ScopedView::isEmpty()

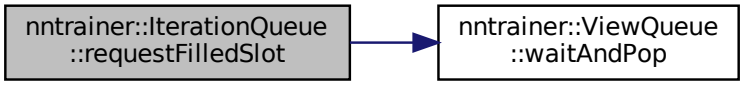
Returns

ScopedView<Iteration> Ieration view. ScopedView::isEmpty() == true if there is no more data coming. Destroying the returned object will signal the queue that the sample is done using.

below is useful information when debugging iteration queue, but there will be too much log if we turn the log on. so leaving it as a comment for now.

Definition at line 93 of file iteration_queue.cpp.

Here is the call graph for this function:



8.221.3.5 slots()

```
unsigned int nntrainer::IterationQueue::slots ( )  [inline]
```

get slot size, slot size is number of batches inside the queue

Returns

unsigned int num slot

Definition at line 246 of file iteration_queue.h.

The documentation for this class was generated from the following files:

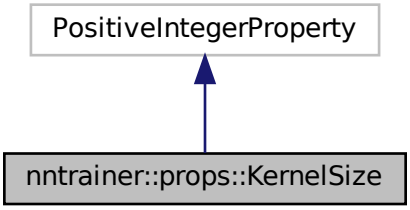
- dataset/iteration_queue.h
- dataset/iteration_queue.cpp

8.222 nntrainer::props::KernelSize Class Reference

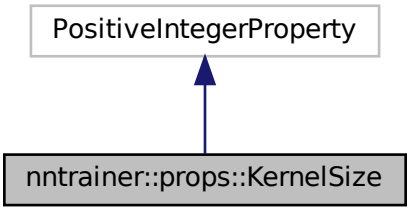
KernelSize property, kernel size is used to measure the filter size.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::KernelSize:



Collaboration diagram for nntrainer::props::KernelSize:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key = "kernel_size"

8.222.1 Detailed Description

KernelSize property, kernel size is used to measure the filter size.

Definition at line 331 of file common_properties.h.

8.222.2 Member Typedef Documentation

8.222.2.1 prop_tag

```
using nntrainer::props::KernelSize::prop_tag = uint_prop_tag
```

property type

Definition at line 334 of file common_properties.h.

8.222.3 Member Data Documentation

8.222.3.1 key

```
constexpr const char* nntrainer::props::KernelSize::key = "kernel_size"  [static], [constexpr]
```

unique key to access

Definition at line 333 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.223 tflite::L2NormOptionsBuilder Struct Reference

Public Types

- typedef L2NormOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- **L2NormOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- L2NormOptionsBuilder & **operator=** (const L2NormOptionsBuilder &)
- flatbuffers::Offset< L2NormOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.223.1 Detailed Description

Definition at line 3605 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

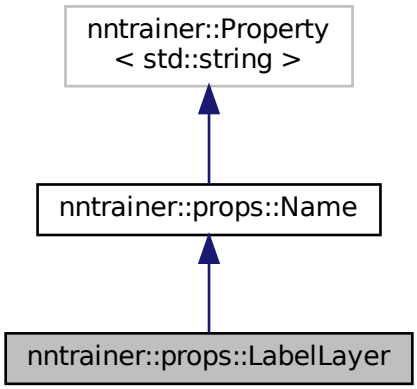
- compiler/tf_schema_generated.h

8.224 nntrainer::props::LabelLayer Class Reference

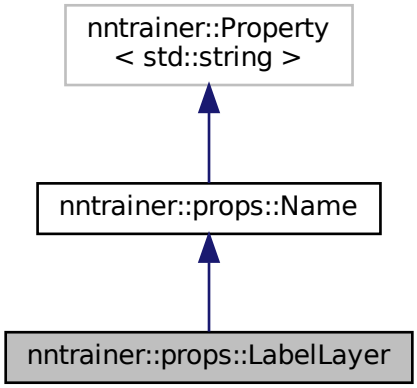
label Layer name property which saves a single connection (practically, std::vector<LabelLayers> is used)

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::LabelLayer:



Collaboration diagram for nntrainer::props::LabelLayer:



Public Types

- using **prop_tag** = str_prop_tag

Public Member Functions

- LabelLayer ()

  Construct LabelLayer object.
- LabelLayer (const std::string &name)

  Construct LabelLayer with the given name.

Static Public Attributes

- static constexpr const char * **key** = "label_layers"

8.224.1 Detailed Description

label Layer name property which saves a single connection (practically, std::vector<LabelLayers> is used)

Definition at line 818 of file common_properties.h.

8.224.2 Constructor & Destructor Documentation

8.224.2.1 LabelLayer() [1/2]

```
nntrainer::props::LabelLayer::LabelLayer ( )
```

Construct LabelLayer object.

Definition at line 286 of file common_properties.cpp.

8.224.2.2 LabelLayer() [2/2]

```
nntrainer::props::LabelLayer::LabelLayer (
            const std::string & name )
```

Construct LabelLayer with the given name.

Parameters

name	Name for the input_layers

Definition at line 287 of file common_properties.cpp.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.225 Layer Class Reference

Class for Layer context.

8.225.1 Detailed Description

Class for Layer context.

for all layers

This provides for the layer initialization. This context will not contain any structures which allow allocation of memory or support to allocate any new memory, but rather only support storing specifications based on which memory will be allocated later.

Definition at line 27 of file layer_context.h.

The documentation for this class was generated from the following file:

- layers/layer_context.h

8.226 Layer Class Reference

Class for Layer context.

8.226.1 Detailed Description

Class for Layer context.

for all layers

This provides for the layer initialization. This context will not contain any structures which allow allocation of memory or support to allocate any new memory, but rather only support storing specifications based on which memory will be allocated later.

Definition at line 27 of file layer_context.h.

The documentation for this class was generated from the following file:

- layers/layer_context.h

8.227 Layer Class Reference

Class for Layer context.

8.227.1 Detailed Description

Class for Layer context.

for all layers

This provides for the layer initialization. This context will not contain any structures which allow allocation of memory or support to allocate any new memory, but rather only support storing specifications based on which memory will be allocated later.

Definition at line 27 of file layer_context.h.

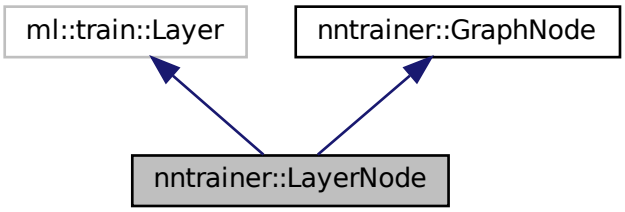
The documentation for this class was generated from the following file:

- layers/layer_context.h

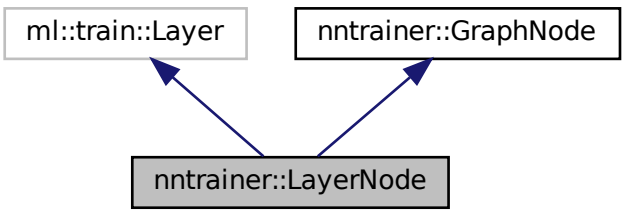
8.228 nntrainer::LayerNode Class Reference

layer node class for the graph

Inheritance diagram for nntrainer::LayerNode:



Collaboration diagram for nntrainer::LayerNode:



Public Types

- enum PrintPreset { PrintPreset::PRINT_NONE = 0, PrintPreset::PRINT_SUMMARY, PrintPreset::PRINT_SUMMARY_META, PrintPreset::PRINT_ALL }

  Preset modes for printing summary for the layer.

Public Member Functions

- LayerNode (std::unique_ptr< nntrainer::Layer > &&l)

  Constructor of LayerNode class for v2.
- ~LayerNode ()

  Destructor of LayerNode Class.
- const std::string getType () const override

  Get the layer type.
- void setProperty (const std::vector< std::string > &properties) override

  set Property of layer
- const std::string getName () const noexcept override

  Get name of the layer.
- void setName (const std::string &name) override

  set name of layer
- const unsigned getInputConnectionIndex (unsigned nth) const

  Get the Input Connection Index object.
- const std::string & getInputConnectionName (unsigned nth) const

  Get the Input Connection Name object.
- void setInputConnectionIndex (unsigned nth, unsigned index)

  Set the Input Connection Index object.
- void setInputConnectionName (unsigned nth, const std::string &name)

  Get the Input Connection Name object.
- const Connection * getOutputConnection (unsigned nth) const

  Get the output connection object.
- void setOutputConnection (unsigned nth, const std::string &name, unsigned index)

  Set the Output Connection.
- const std::vector< std::string > getInputConnections () const override

  Get the input connections for this node.
- const std::vector< std::string > getOutputConnections () const override

  Get the output connections for this node.
- ExecutionOrder getExecutionOrder () const override

  get the execution order/location of this node
- void setExecutionOrder (ExecutionOrder exec_order_) override

  set the execution order/location of this node
- InitLayerContext finalize (const std::vector< TensorDim > &input_dims={})

  Finalize creating the layer node.
- void forwarding (bool training=true)

  Forward Propagation of a layer.
- void calcDerivative ()

  calc the derivative to be passed to the previous layer
- void calcGradient ()

  Calculate the derivative of a layer.
- void exportTo (Exporter &exporter, const ml::train::ExportMethods &method) const

this function helps exporting the layer in a predefined format, while workarounding issue caused by templated function type eraser

- void setBatch (unsigned int batch)

  Set the batch for the layer.
- bool supportInPlace () const

  If the current layer can support in-place.
- void executeInPlace (InPlace val)

  Notify that this layer will execute in-place.
- InPlace executeInPlace () const

  Get if the layer is going to execute in-place.
- bool requireLabel () const

  check if this layer requires label to be passed
- bool supportBackwarding () const

  Get the trainable property of the underlying object.
- bool getTrainable () const

  Get the trainable property of the underlying object.
- bool getFlatten () const

  get if the output of this layer must be flatten
- std::string getSharedFrom () const

  Get the Shared From property of the layer node.
- bool getDistribute () const

  get distribute for this layer
- ActivationType getActivationToBeRealized () const

  get activation for this layer
- ActivationType getActivationType () const

  Activation Type Getter.
- unsigned int getNumInputConnections () const

  Get number of input connections.
- unsigned int getNumOutputConnections () const

  Get number of output connections.
- unsigned int getNumInputs () const

  Get number of inputs.
- unsigned int getNumOutputs () const

  Get number of outputs.
- unsigned int getNumWeights () const

  Get the number of weights.
- void setOutputLayers (const std::vector< std::string > &layers)

  Set the Output Layers object.
- bool hasInputShapeProperty () const

  check if input shape property is set
- const std::vector< TensorDim > getInputDimensions () const

  Get the input dimension.
- const std::vector< TensorDim > getOutputDimensions () const

  Get the output dimension.
- Weight getWeightWrapper (unsigned int idx)

  Get the Weight object.
- Weight & getWeightObject (unsigned int idx)

  Get the Weight object.
- Tensor & getWeight (unsigned int idx)

  Get the Weight tensor object.
- Tensor & getWeightGrad (unsigned int idx)

Get the Weight Gradient tensor object.

- const std::string & getWeightName (unsigned int idx) override

  Get the Weight object name.
- const std::vector< float * > getWeights () override

  Get weight data of the layer.
- void getWeights (std::vector< float * > &weights, std::vector< TensorDim > &weight_dim) override

  Get weight data of the layer.
- void setWeights (const std::vector< float * > weights) override

  Set weight data of the layer.
- Tensor & getInput (unsigned int idx)

  Get the Input tensor object.
- Tensor & getInputGrad (unsigned int idx)

  Get the Input Grad tensor object.
- Tensor & getOutput (unsigned int idx)

  Get the Output tensor object.
- const Tensor getOutputGrad (unsigned int idx) const

  Get the Output Grad tensor object.
- const Tensor & getOutputGradUnsafe (unsigned int idx) const

  Get the Output Grad unsafe.
- void read (std::ifstream &file, bool opt_var=false)

  read layer Weight & Bias data from file
- void save (std::ofstream &file, bool opt_var=false) const

  save layer Weight & Bias data from file
- void clearOptVar ()

  clear optimizer variable to initial state
- float getLoss () const

  get loss for the layer
- RunLayerContext & getRunContext ()

  Get run layer context.
- const RunLayerContext & getRunContext () const

  Get run layer context.
- bool isFinalized () const

  check if layer is finalized
- void configureRunContext (const std::vector< Weight * > &weights, const std::vector< Var_Grad * > &inputs, const std::vector< Var_Grad * > &outputs, const std::vector< Var_Grad * > &tensors)

  Set the Run Context object with given tensor packs.
- void printPreset (std::ostream &out, PrintPreset preset=PrintPreset::PRINT_SUMMARY)

  print using PrintPreset
- void remapIdentifiers (std::function< void(std::string &)> remap_fn)

  remap identifier inside layer node
- void remapConnections (std::function< void(std::string &, unsigned &)> remap_fn)

  remap connections(input, output layers) inside layer node
- std::unique_ptr< LayerNode > cloneConfiguration ()

  create the same node with same properties and types
- void needsCalcDerivative (bool nb)

  Set if the layer needs to do derivative calculation.
- void needsCalcGradient (bool nb)

  Set if the layer needs to do calculation of gradients.
- bool needsCalcDerivative ()

  Get the layer needs to do calculation of derivatives.
- bool needsCalcGradient ()

  Set if the layer needs to do calculation of gradient.

Friends

- std::ostream & operator<< (std::ostream &out, const LayerNode &l)

  Overriding output stream for layers and it's derived class.

8.228.1 Detailed Description

layer node class for the graph

Definition at line 71 of file layer_node.h.

8.228.2 Member Enumeration Documentation

8.228.2.1 PrintPreset

```
enum nntrainer::LayerNode::PrintPreset  [strong]
```

Preset modes for printing summary for the layer.

Enumerator

PRINT_NONE	Print nothing
PRINT_SUMMARY	Print preset including summary information
PRINT_SUMMARY_META	Print summary preset that includes meta information
PRINT_ALL	Print everything possible

Definition at line 730 of file layer_node.h.

8.228.3 Constructor & Destructor Documentation

8.228.3.1 LayerNode()

```
nntrainer::LayerNode::LayerNode (
            std::unique_ptr< nntrainer::Layer > && l )
```

Constructor of LayerNode class for v2.

Parameters

l	layer to wrap with, the ownership is transferred to layer node

Definition at line 159 of file layer_node.cpp.

8.228.3.2 ~LayerNode()

```
nntrainer::LayerNode::~LayerNode ( )  [default]
```

Destructor of LayerNode Class.

Destroy the Layer Node object.

8.228.4 Member Function Documentation

8.228.4.1 calcDerivative()

```
void nntrainer::LayerNode::calcDerivative ( )
```

calc the derivative to be passed to the previous layer

context provides access to the weights (if any), inputs, outputs, and tensors (if any) for the layer. Input and output dimensions can be access from the inputs/outputs tensors themselves.

Definition at line 618 of file layer_node.cpp.

8.228.4.2 calcGradient()

```
void nntrainer::LayerNode::calcGradient ( )
```

Calculate the derivative of a layer.

context provides access to the weights (if any), inputs, outputs, and tensors (if any) for the layer. Input and output dimensions can be access from the inputs/outputs tensors themselves.

Definition at line 636 of file layer_node.cpp.

8.228.4.3 clearOptVar()

```
void nntrainer::LayerNode::clearOptVar ( )
```

clear optimizer variable to initial state

Note

read optimizer variables

Definition at line 483 of file layer_node.cpp.

8.228.4.4 cloneConfiguration()

```
std::unique_ptr< LayerNode > nntrainer::LayerNode::cloneConfiguration ( )
```

create the same node with same properties and types

Note

this must be done before finalize() as finalize has some potential to change some properties

Returns

LayerNode newly created node

Definition at line 776 of file layer_node.cpp.

8.228.4.5 configureRunContext()

```
void nntrainer::LayerNode::configureRunContext (
            const std::vector< Weight * > & weights,
            const std::vector< Var_Grad * > & inputs,
            const std::vector< Var_Grad * > & outputs,
            const std::vector< Var_Grad * > & tensors )
```

Set the Run Context object with given tensor packs.

Parameters

weights	weights
inputs	inputs
outputs	outputs
tensors	tensors

Definition at line 688 of file layer_node.cpp.

8.228.4.6 executeInPlace() [1/2]

```
InPlace nntrainer::LayerNode::executeInPlace ( ) const  [inline]
```

Get if the layer is going to execute in-place.

Returns

InPlace type for the layer

Definition at line 315 of file layer_node.h.

8.228.4.7 executeInPlace() [2/2]

```
void nntrainer::LayerNode::executeInPlace (
            InPlace val ) [inline]
```

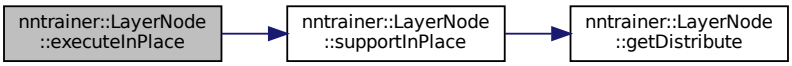
Notify that this layer will execute in-place.

Parameters

val	in place state for the layer

Definition at line 303 of file layer_node.h.

Here is the call graph for this function:



8.228.4.8 exportTo()

```
void nntrainer::LayerNode::exportTo (
            Exporter & exporter,
            const ml::train::ExportMethods & method ) const
```

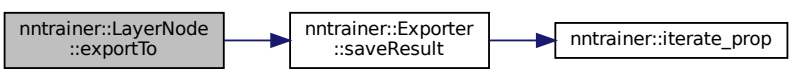
this function helps exporting the layer in a predefined format, while workarounding issue caused by templated function type eraser

Parameters

exporter	exporter that contains exporting logic
method	enum value to identify how it should be exported to

Definition at line 429 of file layer_node.cpp.

Here is the call graph for this function:



8.228.4.9 finalize()

```
InitLayerContext nntrainer::LayerNode::finalize (
            const std::vector< TensorDim > & input_dims = {} )
```

Finalize creating the layer node.

Support all the interface requirements by nntrainer::Layer

Parameters

input_dims	input dimension provided to be used to set output dimensions. if empty function This function must set output dimensions in the given context. Further, context can be used to request weights for the layer, and any extra tensor required for the operation of the layer.

Note

After calling this it is not allowed to change properties.

No memory allocation must be performed in the initialization step. Any tensor memory required must be requested to the context which will be made available during execution of the layer with the context.

configureRunContext() is expected to called right after this.

prepare input dimensions

if prop_dims exist, check if it's same with given input_dims

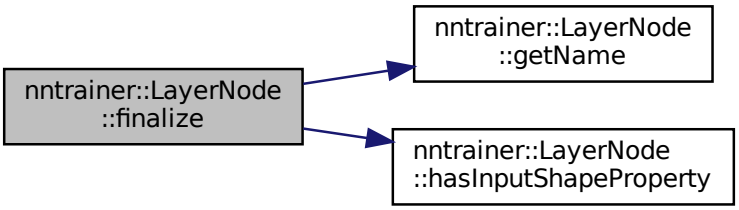
arguably, below check can go away

manipulate layers if required

as we are using output Grad to save label, add fake out info if it's label. This should be substituted to the proper label management

Definition at line 499 of file layer_node.cpp.

Here is the call graph for this function:



8.228.4.10 forwarding()

```
void nntrainer::LayerNode::forwarding (
            bool training = true )
```

Forward Propagation of a layer.

Parameters

training	true if training, false if inference

context provides access to the weights (if any), inputs, outputs, and tensors (if any) for the layer. Input and output dimensions can be access from the inputs/outputs tensors themselves. add loss only for loss layers

Definition at line 596 of file layer_node.cpp.

8.228.4.11 getActivationToBeRealized()

```
ActivationType nntrainer::LayerNode::getActivationToBeRealized ( ) const
```

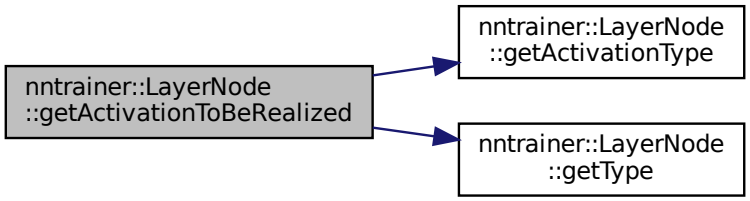
get activation for this layer

Return values

dist	to enable/disable distribute

Definition at line 328 of file layer_node.cpp.

Here is the call graph for this function:



8.228.4.12 getActivationType()

```
ActivationType nntrainer::LayerNode::getActivationType ( ) const
```

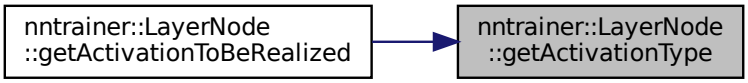
Activation Type Getter.

Return values

Activation	Type.

Definition at line 284 of file layer_node.cpp.

Here is the caller graph for this function:



8.228.4.13 getDistribute()

```
bool nntrainer::LayerNode::getDistribute ( ) const
```

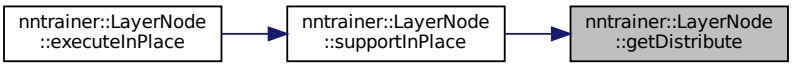
get distribute for this layer

Return values

dist	to enable/disable distribute

Definition at line 362 of file layer_node.cpp.

Here is the caller graph for this function:



8.228.4.14 getExecutionOrder()

```
ExecutionOrder nntrainer::LayerNode::getExecutionOrder ( ) const  [inline], [override], [virtual]
```

get the execution order/location of this node

Return values

the	execution order/location of this node

Implements nntrainer::GraphNode.

Definition at line 217 of file layer_node.h.

8.228.4.15 getFlatten()

```
bool nntrainer::LayerNode::getFlatten ( ) const
```

get if the output of this layer must be flatten

Return values

flatten	value

Definition at line 349 of file layer_node.cpp.

8.228.4.16 getInput()

```
Tensor& nntrainer::LayerNode::getInput (
            unsigned int idx )  [inline]
```

Get the Input tensor object.

Parameters

idx	Identifier of the input

Returns

Tensor& Reference to the input grad tensor

Definition at line 567 of file layer_node.h.

8.228.4.17 getInputConnectionIndex()

```
const unsigned nntrainer::LayerNode::getInputConnectionIndex (
            unsigned nth ) const
```

Get the Input Connection Index object.

Parameters

nth	nth input

Exceptions

if	nth is out of range of getNumInputConnection()

Returns

const unsigned index

Definition at line 207 of file layer_node.cpp.

8.228.4.18 getInputConnectionName()

```
const std::string & nntrainer::LayerNode::getInputConnectionName (
            unsigned nth ) const
```

Get the Input Connection Name object.

Parameters

nth	nth input

Exceptions

if	nth is out of range of getNumInputConnection()

Returns

const std::string& name

Definition at line 213 of file layer_node.cpp.

8.228.4.19 getInputConnections()

```
const std::vector<std::string> nntrainer::LayerNode::getInputConnections ( ) const  [inline],
[override], [virtual]
```

Get the input connections for this node.

Returns

list of name of the nodes which form input connections

Implements nntrainer::GraphNode.

Definition at line 199 of file layer_node.h.

8.228.4.20 getInputDimensions()

```
const std::vector< TensorDim > nntrainer::LayerNode::getInputDimensions ( ) const
```

Get the input dimension.

Returns

TensorDim dimension of the input

Definition at line 401 of file layer_node.cpp.

8.228.4.21 getInputGrad()

```
Tensor& nntrainer::LayerNode::getInputGrad (
            unsigned int idx )  [inline]
```

Get the Input Grad tensor object.

Parameters

idx	Identifier of the input

Returns

Tensor& Reference to the input grad tensor

Definition at line 579 of file layer_node.h.

8.228.4.22 getLoss()

```
float nntrainer::LayerNode::getLoss ( ) const
```

get loss for the layer

Returns

loss of the layer

Definition at line 686 of file layer_node.cpp.

8.228.4.23 getName()

```
const std::string nntrainer::LayerNode::getName ( ) const  [override], [virtual], [noexcept]
```

Get name of the layer.

Return values

name	of the layer

Note

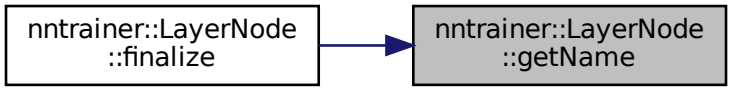
This name is unique to this layer in a model

This name might be changed once this layer is added to the model to keep the name unique to the model

Implements nntrainer::GraphNode.

Definition at line 250 of file layer_node.cpp.

Here is the caller graph for this function:



8.228.4.24 getNumInputConnections()

```
unsigned int nntrainer::LayerNode::getNumInputConnections ( ) const
```

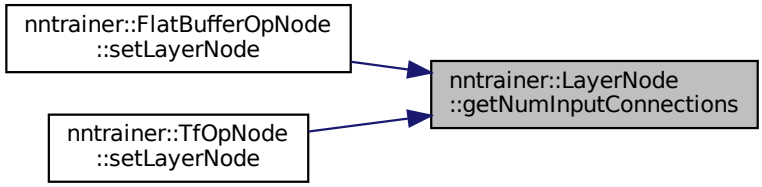
Get number of input connections.

Return values

number	of inputs

Definition at line 293 of file layer_node.cpp.

Here is the caller graph for this function:



8.228.4.25 getNumInputs()

```
unsigned int nntrainer::LayerNode::getNumInputs ( ) const  [inline]
```

Get number of inputs.

Return values

number	of inputs

Definition at line 395 of file layer_node.h.

8.228.4.26 getNumOutputConnections()

```
unsigned int nntrainer::LayerNode::getNumOutputConnections ( ) const
```

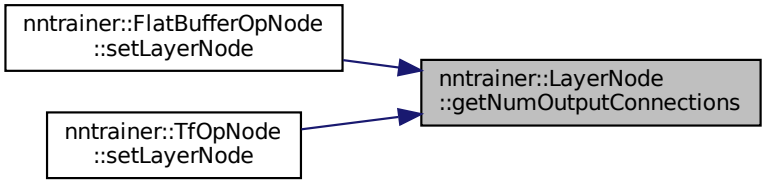
Get number of output connections.

Return values

number	of outputs

Definition at line 299 of file layer_node.cpp.

Here is the caller graph for this function:



8.228.4.27 getNumOutputs()

```
unsigned int nntrainer::LayerNode::getNumOutputs ( ) const  [inline]
```

Get number of outputs.

Return values

number	of outputs

Definition at line 405 of file layer_node.h.

8.228.4.28 getNumWeights()

```
unsigned int nntrainer::LayerNode::getNumWeights ( ) const  [inline]
```

Get the number of weights.

Returns

unsigned int number of weights

Definition at line 416 of file layer_node.h.

8.228.4.29 getOutput()

```
Tensor& nntrainer::LayerNode::getOutput (
            unsigned int idx )  [inline]
```

Get the Output tensor object.

Parameters

idx	Identifier of the output

Returns

Tensor& Reference to the output tensor

Definition at line 591 of file layer_node.h.

8.228.4.30 getOutputConnection()

```
const Connection * nntrainer::LayerNode::getOutputConnection (
            unsigned nth ) const
```

Get the output connection object.

Parameters

nth	nth input

Exceptions

if	nth is out of range of getNumOutputConnection()

Returns

Connection * view of a connection, null means this does not exist

Definition at line 231 of file layer_node.cpp.

8.228.4.31 getOutputConnections()

```
const std::vector<std::string> nntrainer::LayerNode::getOutputConnections ( ) const  [inline],
[override], [virtual]
```

Get the output connections for this node.

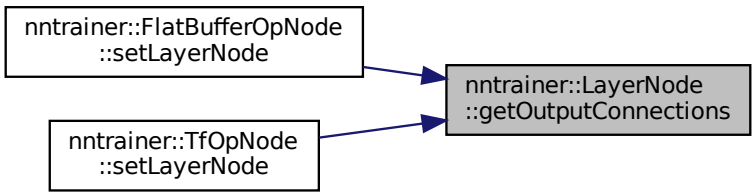
Returns

list of name of the nodes which form output connections

Implements nntrainer::GraphNode.

Definition at line 208 of file layer_node.h.

Here is the caller graph for this function:



8.228.4.32 getOutputDimensions()

```
const std::vector< TensorDim > nntrainer::LayerNode::getOutputDimensions ( ) const
```

Get the output dimension.

Returns

TensorDim dimension of the output

Definition at line 415 of file layer_node.cpp.

8.228.4.33 getOutputGrad()

```
const Tensor nntrainer::LayerNode::getOutputGrad (
            unsigned int idx ) const  [inline]
```

Get the Output Grad tensor object.

Parameters

idx	Identifier of the output

Returns

Tensor& Reference to the output grad tensor

Definition at line 603 of file layer_node.h.

8.228.4.34 getOutputGradUnsafe()

```
const Tensor& nntrainer::LayerNode::getOutputGradUnsafe (
            unsigned int idx ) const  [inline]
```

Get the Output Grad unsafe.

Parameters

idx	Identifier of the output

Returns

Tensor& Reference to the output grad tensor

Definition at line 615 of file layer_node.h.

8.228.4.35 getRunContext() [1/2]

```
RunLayerContext& nntrainer::LayerNode::getRunContext ( )  [inline]
```

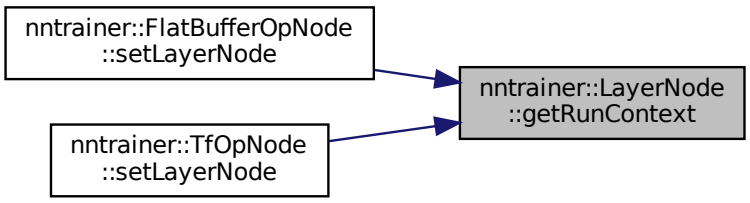
Get run layer context.

Return values

run	layer context

Definition at line 661 of file layer_node.h.

Here is the caller graph for this function:



8.228.4.36 getRunContext() [2/2]

```
const RunLayerContext& nntrainer::LayerNode::getRunContext ( ) const  [inline]
```

Get run layer context.

Return values

run	layer context

Definition at line 672 of file layer_node.h.

8.228.4.37 getSharedFrom()

```
std::string nntrainer::LayerNode::getSharedFrom ( ) const
```

Get the Shared From property of the layer node.

Returns

std::string node name where the weights are borrowed

Definition at line 357 of file layer_node.cpp.

8.228.4.38 getTrainable()

```
bool nntrainer::LayerNode::getTrainable ( ) const  [virtual]
```

Get the trainable property of the underlying object.

Support interfaces for the properties intercepted from layer

Returns

boolean true if trainable, else false

if a layer does not contain any weights, it will be treated as a non-trainable layer.

Implements nntrainer::GraphNode.

Definition at line 337 of file layer_node.cpp.

8.228.4.39 getType()

```
const std::string nntrainer::LayerNode::getType ( ) const  [override], [virtual]
```

Get the layer type.

Support all the interface requirements by ml::train::Layer

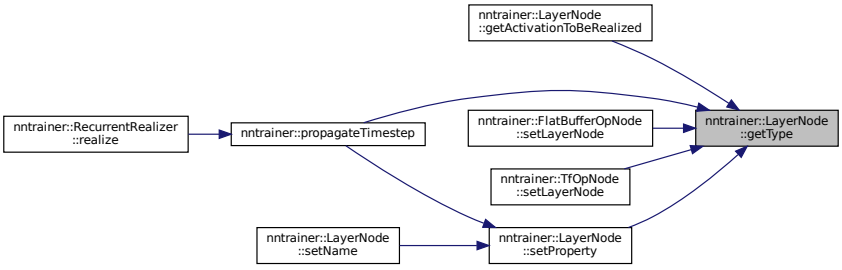
Returns

const std::string type representation

Implements nntrainer::GraphNode.

Definition at line 335 of file layer_node.cpp.

Here is the caller graph for this function:



8.228.4.40 getWeight()

```
Tensor& nntrainer::LayerNode::getWeight (
            unsigned int idx )  [inline]
```

Get the Weight tensor object.

Parameters

idx	Identifier of the weight

Returns

Tensor& Reference to the weight tensor

Definition at line 484 of file layer_node.h.

8.228.4.41 getWeightGrad()

```
Tensor& nntrainer::LayerNode::getWeightGrad (
            unsigned int idx )  [inline]
```

Get the Weight Gradient tensor object.

Parameters

idx	Identifier of the weight

Returns

Tensor& Reference to the weight grad tensor

Definition at line 496 of file layer_node.h.

8.228.4.42 getWeightName()

```
const std::string& nntrainer::LayerNode::getWeightName (
            unsigned int idx )  [inline], [override]
```

Get the Weight object name.

Parameters

idx	Identifier of the weight

Returns

const std::string &Name of the weight

Definition at line 508 of file layer_node.h.

8.228.4.43 getWeightObject()

```
Weight& nntrainer::LayerNode::getWeightObject (
            unsigned int idx )  [inline]
```

Get the Weight object.

Parameters

idx	Identifier of the weight

Returns

Tensor& Reference to the weight tensor

Definition at line 472 of file layer_node.h.

8.228.4.44 getWeights() [1/2]

```
const std::vector<float *> nntrainer::LayerNode::getWeights ( )  [inline], [override]
```

Get weight data of the layer.

Return values

weight	data of the layer

Note

nntrainer assign the vector and if there is no weights, the size of vector is zero

layer needs to be finalized before called.

Definition at line 521 of file layer_node.h.

8.228.4.45 getWeights() [2/2]

```
void nntrainer::LayerNode::getWeights (
            std::vector< float * > & weights,
            std::vector< TensorDim > & weight_dim )  [inline], [override]
```

Get weight data of the layer.

Parameters

out	*weights*	: float * arrary to store weight data
out	*weights_dim*	: TensorDim for each weights

Note

nntrainer assign the vector and if there is no weights, the size of vector is zero

layer needs to be finalized before called.

Definition at line 540 of file layer_node.h.

8.228.4.46 getWeightWrapper()

```
Weight nntrainer::LayerNode::getWeightWrapper (
            unsigned int idx )  [inline]
```

Get the Weight object.

Parameters

idx	Identifier of the weight

Returns

Weight& Reference to the weight

Definition at line 453 of file layer_node.h.

8.228.4.47 hasInputShapeProperty()

```
bool nntrainer::LayerNode::hasInputShapeProperty ( ) const
```

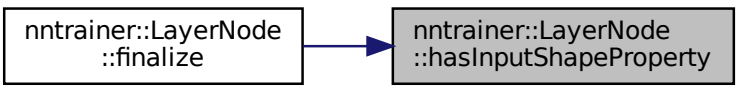
check if input shape property is set

Returns

bool true if input shape property has set

Definition at line 392 of file layer_node.cpp.

Here is the caller graph for this function:



8.228.4.48 isFinalized()

```
bool nntrainer::LayerNode::isFinalized ( ) const  [inline]
```

check if layer is finalized

Return values

bool	true if the layer is finalized else false

Definition at line 707 of file layer_node.h.

8.228.4.49 needsCalcDerivative() [1/2]

```
bool nntrainer::LayerNode::needsCalcDerivative ( )  [inline]
```

Get the layer needs to do calculation of derivatives.

Returns

true if the layer needs to do backwarding, else false

Definition at line 795 of file layer_node.h.

8.228.4.50 needsCalcDerivative() [2/2]

```
void nntrainer::LayerNode::needsCalcDerivative (
            bool nb )  [inline]
```

Set if the layer needs to do derivative calculation.

Parameters

nb	true if the layer needs to do backwarding, else false

Definition at line 776 of file layer_node.h.

8.228.4.51 needsCalcGradient() [1/2]

```
bool nntrainer::LayerNode::needsCalcGradient ( )  [inline]
```

Set if the layer needs to do calculation of gradient.

Parameters

nb	true if the layer needs to do backwarding, else false

Definition at line 802 of file layer_node.h.

8.228.4.52 needsCalcGradient() [2/2]

```
void nntrainer::LayerNode::needsCalcGradient (
            bool nb )  [inline]
```

Set if the layer needs to do calculation of gradients.

Parameters

nb	true if the layer needs to do backwarding, else false

Definition at line 788 of file layer_node.h.

8.228.4.53 printPreset()

```
void nntrainer::LayerNode::printPreset (
            std::ostream & out,
            PrintPreset preset = PrintPreset::PRINT_SUMMARY )
```

print using PrintPreset

Parameters

out	oustream
preset	preset to be used

fall through intended

fall through intended

Definition at line 712 of file layer_node.cpp.

8.228.4.54 read()

```
void nntrainer::LayerNode::read (
            std::ifstream & file,
            bool opt_var = false )
```

read layer Weight & Bias data from file

Parameters

file	input file stream
bool	read optimizer variables

Note

read optimizer variables

shared weights are only be read at the first acecss

Definition at line 435 of file layer_node.cpp.

8.228.4.55 remapConnections()

```
void nntrainer::LayerNode::remapConnections (
            std::function< void(std::string &, unsigned &)> remap_fn )
```

remap connections(input, output layers) inside layer node

Parameters

remap_fn	function to remap

Definition at line 751 of file layer_node.cpp.

8.228.4.56 remapIdentifiers()

```
void nntrainer::LayerNode::remapIdentifiers (
            std::function< void(std::string &)> remap_fn )
```

remap identifier inside layer node

Parameters

remap_fn	function to remap

remap connections without touching index

Definition at line 733 of file layer_node.cpp.

8.228.4.57 requireLabel()

```
bool nntrainer::LayerNode::requireLabel ( ) const
```

check if this layer requires label to be passed

Returns

true if requires a label when training, else false

Note

if requireLabel() == true means, for now, that it is endpoint of a graph(numOutlayers == 0). label will be fed to the gradient of hidden if requireLabel is true

Definition at line 680 of file layer_node.cpp.

8.228.4.58 save()

```
void nntrainer::LayerNode::save (
            std::ofstream & file,
            bool opt_var = false ) const
```

save layer Weight & Bias data from file

Parameters

file	output file stream
bool	save optimizer variables

Definition at line 457 of file layer_node.cpp.

8.228.4.59 setBatch()

```
void nntrainer::LayerNode::setBatch (
            unsigned int batch )
```

Set the batch for the layer.

Parameters

batch	Batch value to be set

Update the run context based on the updated batch size if required

Definition at line 656 of file layer_node.cpp.

8.228.4.60 setExecutionOrder()

```
void nntrainer::LayerNode::setExecutionOrder (
            ExecutionOrder exec_order_ )  [inline], [override], [virtual]
```

set the execution order/location of this node

Parameters

exec_order	the execution order/location of this node

Implements nntrainer::GraphNode.

Definition at line 224 of file layer_node.h.

8.228.4.61 setInputConnectionIndex()

```
void nntrainer::LayerNode::setInputConnectionIndex (
            unsigned nth,
            unsigned index )
```

Set the Input Connection Index object.

Parameters

nth	nth input
index	index to set

Exceptions

if	nth is out of range of getNumInputConnection()

Definition at line 219 of file layer_node.cpp.

8.228.4.62 setInputConnectionName()

```
void nntrainer::LayerNode::setInputConnectionName (
            unsigned nth,
            const std::string & name )
```

Get the Input Connection Name object.

Parameters

nth	input
index	index to set

Exceptions

if	nth is out of range of getNumInputConnection()
if	new identifier is invalid

Definition at line 225 of file layer_node.cpp.

8.228.4.63 setName()

```
void nntrainer::LayerNode::setName (
            const std::string & name )  [inline], [override], [virtual]
```

set name of layer

Support all the interface requirements by nntrainer::GraphNode

Parameters

in	*name*	Name of the layer

Implements nntrainer::GraphNode.

Definition at line 127 of file layer_node.h.

Here is the call graph for this function:

8.228.4.64 setOutputConnection()

```
void nntrainer::LayerNode::setOutputConnection (
            unsigned nth,
            const std::string & name,
            unsigned index )
```

Set the Output Connection.

Note

Each output must be identified only ONCE.

when nth comes, getNumOutput() expends to nth + 1 as resize occurs. Please also notice none identified intermediate output (or mismatch between actual number of out tensors and output) is allowed but will produce warning, this implies that the output is not used else where.

Exceptions

std::invalid_argument	when trying to identify output more then once

Parameters

nth	nth output
name	name of the output bound connection
index	index of the output bound connection

Definition at line 235 of file layer_node.cpp.

8.228.4.65 setOutputLayers()

```
void nntrainer::LayerNode::setOutputLayers (
            const std::vector< std::string > & layers )
```

Set the Output Layers object.

Parameters

layers	Name of the layers

Definition at line 384 of file layer_node.cpp.

8.228.4.66 setProperty()

```
void nntrainer::LayerNode::setProperty (
            const std::vector< std::string > & properties )  [override]
```

set Property of layer

Parameters

in	*properties*	values of property

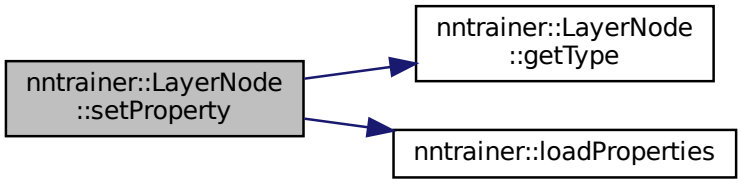
Return values

ML_ERROR_NONE	Successful.
ML_ERROR_INVALID_PARAMETER	invalid parameter.

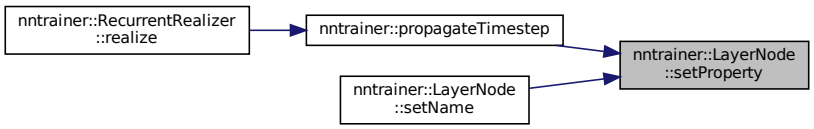
This function accepts vector of properties in the format - { std::string property_name=property_val, ...}

Definition at line 179 of file layer_node.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.228.4.67 setWeights()

```
void nntrainer::LayerNode::setWeights (
            const std::vector< float * > weights )  [override]
```

Set weight data of the layer.

Note

Size of vector must be the same with number of weights.

layer needs to be finalized before called.

Definition at line 193 of file layer_node.cpp.

8.228.4.68 supportBackwarding()

```
bool nntrainer::LayerNode::supportBackwarding ( ) const  [inline]
```

Get the trainable property of the underlying object.

Add rest of the helper interfaces required by other internal classes

Returns

boolean true if trainable, else false

Definition at line 335 of file layer_node.h.

8.228.4.69 supportInPlace()

```
bool nntrainer::LayerNode::supportInPlace ( ) const
```

If the current layer can support in-place.

Returns

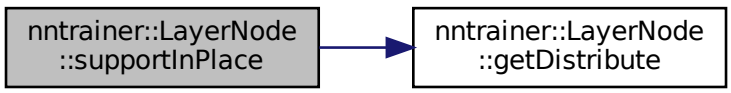
true if inplace, else false

Note

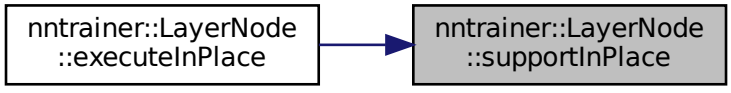
below is a quick fix, we need to have a guard that this shouldn't be query until realizeProps has been finalized (which means we will need another end point to fixate this property)

Definition at line 666 of file layer_node.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- layers/layer_node.h
- layers/layer_node.cpp

8.229 tflite::LeakyReluOptionsBuilder Struct Reference

Public Types

- typedef LeakyReluOptions **Table**

Public Member Functions

- void **add_alpha** (float alpha)
- **LeakyReluOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LeakyReluOptionsBuilder & **operator=** (const LeakyReluOptionsBuilder &)
- flatbuffers::Offset< LeakyReluOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.229.1 Detailed Description

Definition at line 6307 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

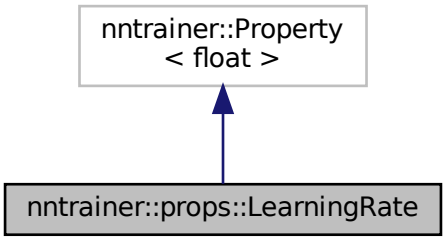
- compiler/tf_schema_generated.h

8.230 nntrainer::props::LearningRate Class Reference

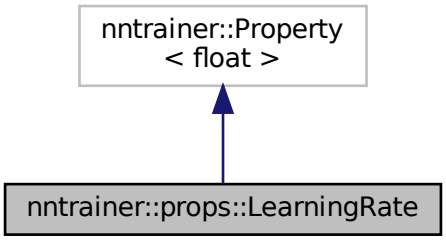
Learning Rate props.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::LearningRate:



Collaboration diagram for nntrainer::props::LearningRate:



Public Types

- using prop_tag = float_prop_tag

Static Public Attributes

- static constexpr const char * key

8.230.1 Detailed Description

Learning Rate props.

Definition at line 1284 of file common_properties.h.

8.230.2 Member Typedef Documentation

8.230.2.1 prop_tag

```
using nntrainer::props::LearningRate::prop_tag = float_prop_tag
```

property type

Definition at line 1288 of file common_properties.h.

8.230.3 Member Data Documentation

8.230.3.1 key

```
constexpr const char* nntrainer::props::LearningRate::key  [static], [constexpr]
```

Initial value:

```
=
    "learning_rate"
```

unique key to access

Definition at line 1286 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.231 tflite::LessEqualOptionsBuilder Struct Reference

Public Types

- typedef LessEqualOptions **Table**

Public Member Functions

- **LessEqualOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LessEqualOptionsBuilder & **operator=** (const LessEqualOptionsBuilder &)
- flatbuffers::Offset< LessEqualOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.231.1 Detailed Description

Definition at line 5336 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.232 tflite::LessOptionsBuilder Struct Reference

Public Types

- typedef LessOptions **Table**

Public Member Functions

- **LessOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LessOptionsBuilder & **operator=** (const LessOptionsBuilder &)
- flatbuffers::Offset< LessOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.232.1 Detailed Description

Definition at line 5306 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.233 tflite::LocalResponseNormalizationOptionsBuilder Struct Reference

Public Types

- typedef LocalResponseNormalizationOptions **Table**

Public Member Functions

- void **add_radius** (int32_t radius)
- void **add_bias** (float bias)
- void **add_alpha** (float alpha)
- void **add_beta** (float beta)
- **LocalResponseNormalizationOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LocalResponseNormalizationOptionsBuilder & **operator=** (const LocalResponseNormalizationOptionsBuilder &)
- flatbuffers::Offset< LocalResponseNormalizationOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.233.1 Detailed Description

Definition at line 3662 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.234 tflite::LogicalAndOptionsBuilder Struct Reference

Public Types

- typedef LogicalAndOptions **Table**

Public Member Functions

- **LogicalAndOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LogicalAndOptionsBuilder & **operator=** (const LogicalAndOptionsBuilder &)
- flatbuffers::Offset< LogicalAndOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.234.1 Detailed Description

Definition at line 6008 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.235 tflite::LogicalNotOptionsBuilder Struct Reference

Public Types

- typedef LogicalNotOptions **Table**

Public Member Functions

- **LogicalNotOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LogicalNotOptionsBuilder & **operator=** (const LogicalNotOptionsBuilder &)
- flatbuffers::Offset< LogicalNotOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.235.1 Detailed Description

Definition at line 6038 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.236 tflite::LogicalOrOptionsBuilder Struct Reference

Public Types

- typedef LogicalOrOptions **Table**

Public Member Functions

- **LogicalOrOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LogicalOrOptionsBuilder & **operator=** (const LogicalOrOptionsBuilder &)
- flatbuffers::Offset< LogicalOrOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.236.1 Detailed Description

Definition at line 5876 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.237 tflite::LogSoftmaxOptionsBuilder Struct Reference

Public Types

- typedef LogSoftmaxOptions **Table**

Public Member Functions

- **LogSoftmaxOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LogSoftmaxOptionsBuilder & **operator=** (const LogSoftmaxOptionsBuilder &)
- flatbuffers::Offset< LogSoftmaxOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.237.1 Detailed Description

Definition at line 4990 of file tf_schema_generated.h.

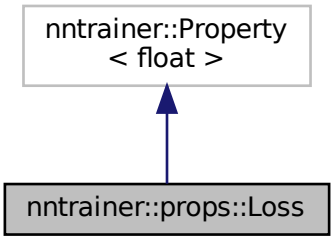
The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

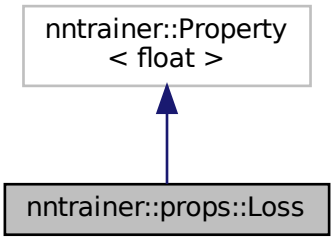
8.238 nntrainer::props::Loss Class Reference

Loss property, this defines loss specification of layer.

Inheritance diagram for nntrainer::props::Loss:



Collaboration diagram for nntrainer::props::Loss:



Public Types

- using prop_tag = float_prop_tag

Public Member Functions

- Loss (float value=0.0)

  Construct a new loss object with a default value 0.0.
- bool isValid (const float &v) const override

  LossSpec validator.

Static Public Attributes

- static constexpr const char * key = "loss"

8.238.1 Detailed Description

Loss property, this defines loss specification of layer.

Definition at line 69 of file layer_node.cpp.

8.238.2 Member Typedef Documentation

8.238.2.1 prop_tag

```
using nntrainer::props::Loss::prop_tag = float_prop_tag
```

property type

Definition at line 78 of file layer_node.cpp.

8.238.3 Constructor & Destructor Documentation

8.238.3.1 Loss()

```
nntrainer::props::Loss::Loss (
            float value = 0.0 )  [inline]
```

Construct a new loss object with a default value 0.0.

Definition at line 76 of file layer_node.cpp.

8.238.4 Member Function Documentation

8.238.4.1 isValid()

```
bool nntrainer::props::Loss::isValid (
            const float & v ) const  [inline], [override]
```

LossSpec validator.

Todo detect when loss becomes Nan is useful. But it will need dedicated throw

Parameters

v	float to validate

Return values

true	if is valid number
false	if it is nan

Definition at line 88 of file layer_node.cpp.

8.238.5 Member Data Documentation

8.238.5.1 key

```
constexpr const char* nntrainer::props::Loss::key = "loss"  [static], [constexpr]
```

unique key to access

Definition at line 77 of file layer_node.cpp.

The documentation for this class was generated from the following file:

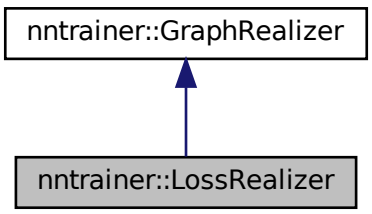
- layers/layer_node.cpp

8.239 nntrainer::LossRealizer Class Reference

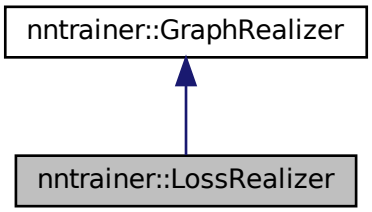
Graph realizer class which removes loss layer from the graph.

```
#include <loss_realizer.h>
```

Inheritance diagram for nntrainer::LossRealizer:



Collaboration diagram for nntrainer::LossRealizer:



Public Member Functions

- LossRealizer ()=default

  Construct a new Loss Realizer object.
- ~LossRealizer ()=default

  Destroy the Graph Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  graph realizer creates a shallow copied graph based on the reference

8.239.1 Detailed Description

Graph realizer class which removes loss layer from the graph.

Note

This assumes the number of input / output connection of loss layer == 1

Definition at line 26 of file loss_realizer.h.

8.239.2 Constructor & Destructor Documentation

8.239.2.1 LossRealizer()

```
nntrainer::LossRealizer::LossRealizer ( )  [default]
```

Construct a new Loss Realizer object.

8.239.2.2 ~LossRealizer()

```
nntrainer::LossRealizer::~LossRealizer ( )  [default]
```

Destroy the Graph Realizer object.

8.239.3 Member Function Documentation

8.239.3.1 realize()

```
GraphRepresentation nntrainer::LossRealizer::realize (
            const GraphRepresentation & reference )  [override], [virtual]
```

graph realizer creates a shallow copied graph based on the reference

Note

loss realizer removes loss layers from GraphRepresentation

Parameters

reference	GraphRepresentation to be realized

Todo support more loss layers

Note

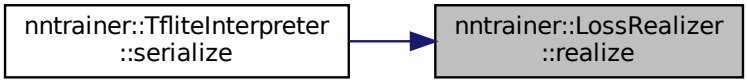
Some layers need to consider not removing all semantics. For example, When CrossEntropySigmoidLoss↩ Layer needs to be removed, sigmoid computation shouldn't be removed.

Assume that loss layers don't have output connections

Implements nntrainer::GraphRealizer.

Definition at line 29 of file loss_realizer.cpp.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- compiler/loss_realizer.h
- compiler/loss_realizer.cpp

8.240 tflite::LSHProjectionOptionsBuilder Struct Reference

Public Types

- typedef LSHProjectionOptions **Table**

Public Member Functions

- void **add_type** (tflite::LSHProjectionType type)
- **LSHProjectionOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LSHProjectionOptionsBuilder & **operator=** (const LSHProjectionOptionsBuilder &)
- flatbuffers::Offset< LSHProjectionOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.240.1 Detailed Description

Definition at line 3055 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.241 tflite::LSTMOptionsBuilder Struct Reference

Public Types

- typedef LSTMOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_cell_clip** (float cell_clip)
- void **add_proj_clip** (float proj_clip)
- void **add_kernel_type** (tflite::LSTMKernelType kernel_type)
- void **add_asymmetric_quantize_inputs** (bool asymmetric_quantize_inputs)
- **LSTMOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- LSTMOptionsBuilder & **operator=** (const LSTMOptionsBuilder &)
- flatbuffers::Offset< LSTMOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.241.1 Detailed Description

Definition at line 3739 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.242 tflite::MatrixDiagOptionsBuilder Struct Reference

Public Types

- typedef MatrixDiagOptions **Table**

Public Member Functions

- **MatrixDiagOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- MatrixDiagOptionsBuilder & **operator=** (const MatrixDiagOptionsBuilder &)
- flatbuffers::Offset< MatrixDiagOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.242.1 Detailed Description

Definition at line 6628 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.243 tflite::MatrixSetDiagOptionsBuilder Struct Reference

Public Types

- typedef MatrixSetDiagOptions **Table**

Public Member Functions

- **MatrixSetDiagOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- MatrixSetDiagOptionsBuilder & **operator=** (const MatrixSetDiagOptionsBuilder &)
- flatbuffers::Offset< MatrixSetDiagOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.243.1 Detailed Description

Definition at line 6688 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.244 tflite::MaximumMinimumOptionsBuilder Struct Reference

Public Types

- typedef MaximumMinimumOptions **Table**

Public Member Functions

- **MaximumMinimumOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- MaximumMinimumOptionsBuilder & **operator=** (const MaximumMinimumOptionsBuilder &)
- flatbuffers::Offset< MaximumMinimumOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.244.1 Detailed Description

Definition at line 5102 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

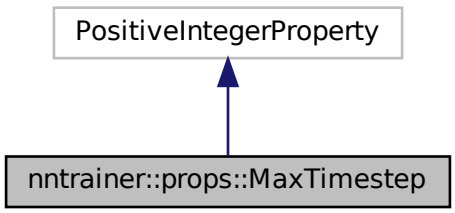
- compiler/tf_schema_generated.h

8.245 nntrainer::props::MaxTimestep Class Reference

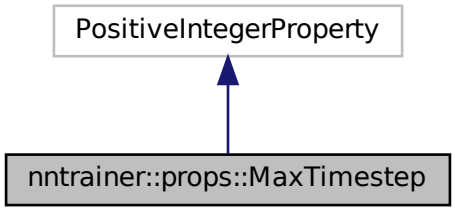
maximum timestep property, timestep is used to identify for the maximum time unroll possible for lstm/gru/rnn layer

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::MaxTimestep:



Collaboration diagram for nntrainer::props::MaxTimestep:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key

8.245.1 Detailed Description

maximum timestep property, timestep is used to identify for the maximum time unroll possible for lstm/gru/rnn layer

Definition at line 1116 of file common_properties.h.

8.245.2 Member Typedef Documentation

8.245.2.1 prop_tag

```
using nntrainer::props::MaxTimestep::prop_tag = uint_prop_tag
```

property type

Definition at line 1120 of file common_properties.h.

8.245.3 Member Data Documentation

8.245.3.1 key

```
constexpr const char* nntrainer::props::MaxTimestep::key  [static], [constexpr]
```

Initial value:

```
=
    "max_timestep"
```

unique key to access

Definition at line 1118 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.246 nntrainer::MemoryData< T > Class Template Reference

MemoryData Class.

```
#include <memory_data.h>
```

Public Member Functions

- MemoryData (T *addr)
  Constructor of Memory Data.
- MemoryData (unsigned int mem_id, MemoryDataValidateCallback v_cb, MemoryDataValidateCallback i_cb)
  Constructor of Memory Data.
- MemoryData ()=delete
  Deleted constructor of Memory Data.
- MemoryData (MemoryDataValidateCallback v_cb, MemoryDataValidateCallback i_cb)=delete
  Constructor of MemoryData.
- MemoryData (T *addr, MemoryDataValidateCallback v_cb, MemoryDataValidateCallback i_cb)=delete
  Constructor of MemoryData.
- virtual ~MemoryData ()=default
  Destructor of Memory Data.
- void setAddr (T *addr)
  Set address.
- T * getAddr () const
  Get address.
- void validate ()
  Validate memory data.
- void invalidate ()
  Invalidate memory data.
- void setValid (bool v)
  Set valid.

8.246.1 Detailed Description

template<typename T = float>
class nntrainer::MemoryData< T >

MemoryData Class.

Definition at line 26 of file memory_data.h.

8.246.2 Constructor & Destructor Documentation

8.246.2.1 MemoryData() [1/2]

```
template<typename T = float>
nntrainer::MemoryData< T >::MemoryData (
            T * addr )  [inline], [explicit]
```

Constructor of Memory Data.

Parameters

in	*addr*	Memory data

Definition at line 32 of file memory_data.h.

8.246.2.2 MemoryData() [2/2]

```
template<typename T = float>
nntrainer::MemoryData< T >::MemoryData (
            unsigned int mem_id,
            MemoryDataValidateCallback v_cb,
            MemoryDataValidateCallback i_cb )  [inline], [explicit]
```

Constructor of Memory Data.

Parameters

in	*mem_id*	validate callback.
in	*v_cb*	validate callback.
in	*i_cb*	invalidate callback.

Definition at line 45 of file memory_data.h.

The documentation for this class was generated from the following file:

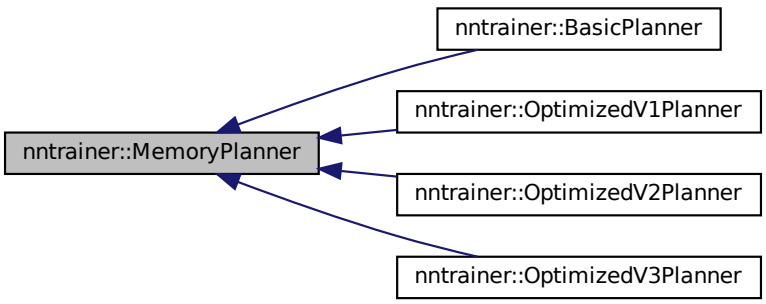
- tensor/memory_data.h

8.247 nntrainer::MemoryPlanner Class Reference

Memory Planner provides the plan/strategy to allocate the memory.

`#include <memory_planner.h>`

Inheritance diagram for nntrainer::MemoryPlanner:



Public Member Functions

- virtual ~MemoryPlanner ()=default

  MemoryPlanner destructor.
- virtual size_t planLayout (const std::vector< size_t > &memory_size, const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, std::vector< size_t > &memory_offset, std::vector< bool > &memory_is_wgrad, size_t n_wgrad) const =0

  Plan the layout for the memory allocation.
- virtual const std::string & getType () const =0

  Get type of the planner.

8.247.1 Detailed Description

Memory Planner provides the plan/strategy to allocate the memory.

Definition at line 26 of file memory_planner.h.

8.247.2 Constructor & Destructor Documentation

8.247.2.1 ~MemoryPlanner()

```
virtual nntrainer::MemoryPlanner::~MemoryPlanner ( )  [virtual], [default]
```

MemoryPlanner destructor.

8.247.3 Member Function Documentation

8.247.3.1 getType()

```
virtual const std::string& nntrainer::MemoryPlanner::getType ( ) const  [pure virtual]
```

Get type of the planner.

Returns

The type of the planner

Implemented in nntrainer::OptimizedV2Planner, nntrainer::OptimizedV1Planner, nntrainer::OptimizedV3Planner, and nntrainer::BasicPlanner.

8.247.3.2 planLayout()

```
virtual size_t nntrainer::MemoryPlanner::planLayout (
            const std::vector< size_t > & memory_size,
            const std::vector< std::pair< unsigned int, unsigned int >> & memory_validity,
            std::vector< size_t > & memory_offset,
            std::vector< bool > & memory_is_wgrad,
            size_t n_wgrad ) const  [pure virtual]
```

Plan the layout for the memory allocation.

Parameters

in	*memory_size*	The size of the various memories
in	*memory_validity*	The validity of the various memories
out	*memory_offset*	The offset of each memory from the base of the allocated memory
in	*memory_is_wgrad*	The index for identification of weight gradient

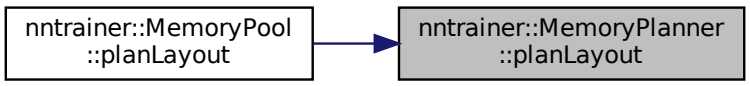
Returns

The total memory required as per this strategy

The minimum offset will be 0, and the maximum offset will be less than the sum of the memory_size vector.

Implemented in nntrainer::OptimizedV2Planner, nntrainer::OptimizedV1Planner, nntrainer::OptimizedV3Planner, and nntrainer::BasicPlanner.

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

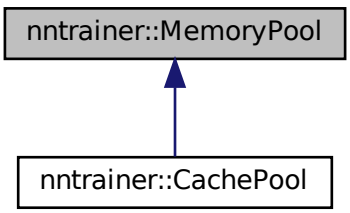
- tensor/memory_planner.h

8.248 nntrainer::MemoryPool Class Reference

Memory Pool provides a common pool for all the tensor memory.

`#include <memory_pool.h>`

Inheritance diagram for nntrainer::MemoryPool:



Public Member Functions

- MemoryPool ()

  MemoryPool default constructor.
- virtual ~MemoryPool ()

  MemoryPool destructor.
- virtual unsigned int requestMemory (size_t bytes, unsigned int start_time, unsigned int end_time, std::vector< unsigned int > exec_order=std::vector< unsigned int >(), TensorLifespan lifespan=TensorLifespan::MAX_LIFESPAN, bool is_wgrad=false)

  Request Memory from memory pool.
- double planLayout (const MemoryPlanner &planner)

  Plan the layout with memory planner.
- virtual void allocate ()

  Do the allocation of memory.
- virtual std::shared_ptr< MemoryData< float > > getMemory (unsigned int idx)

  Get the allocated memory.
- virtual void deallocate ()

  Free all the allocated memory.
- size_t size ()

  Get the maximum real memory requirement.
- size_t minMemoryRequirement ()

  Get the minimum theoretical memory requirement.
- virtual void clear ()

  Clear the memory pool.
- virtual bool isAllocated () const

  Is the memory pool allocated.

Protected Member Functions

- std::vector< size_t > & getMemoryOffset ()

  Get memory offset.
- std::vector< size_t > & getMemorySize ()

  Get memory size.
- std::vector< std::vector< unsigned int > > & getMemoryExecOrder ()

  Get memory execution order.

8.248.1 Detailed Description

Memory Pool provides a common pool for all the tensor memory.

Definition at line 36 of file memory_pool.h.

8.248.2 Constructor & Destructor Documentation

8.248.2.1 MemoryPool()

```
nntrainer::MemoryPool::MemoryPool ( )  [inline], [explicit]
```

MemoryPool default constructor.

Definition at line 42 of file memory_pool.h.

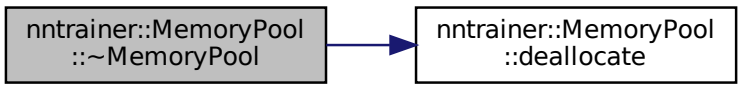
8.248.2.2 ~MemoryPool()

```
virtual nntrainer::MemoryPool::~MemoryPool ( )  [inline], [virtual]
```

MemoryPool destructor.

Definition at line 52 of file memory_pool.h.

Here is the call graph for this function:



8.248.3 Member Function Documentation

8.248.3.1 allocate()

```
void nntrainer::MemoryPool::allocate ( )  [virtual]
```

Do the allocation of memory.

Reimplemented in nntrainer::CachePool.

Definition at line 91 of file memory_pool.cpp.

8.248.3.2 clear()

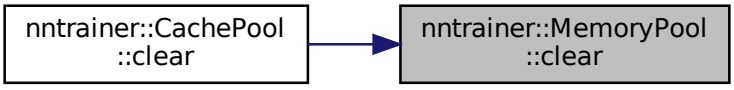
```
void nntrainer::MemoryPool::clear ( )  [virtual]
```

Clear the memory pool.

Reimplemented in nntrainer::CachePool.

Definition at line 317 of file memory_pool.cpp.

Here is the caller graph for this function:



8.248.3.3 deallocate()

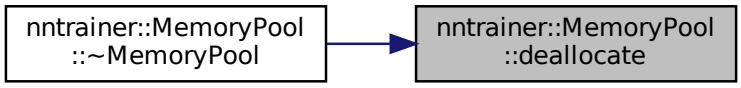
```
void nntrainer::MemoryPool::deallocate ( )  [virtual]
```

Free all the allocated memory.

Reimplemented in nntrainer::CachePool.

Definition at line 130 of file memory_pool.cpp.

Here is the caller graph for this function:



8.248.3.4 getMemory()

```
std::shared_ptr< MemoryData< float > > nntrainer::MemoryPool::getMemory (
            unsigned int idx )  [virtual]
```

Get the allocated memory.

Parameters

token	The token received from the requestMemory

Returns

The pointer of the memory

This function will throw if called before allocation.

Reimplemented in nntrainer::CachePool.

Definition at line 116 of file memory_pool.cpp.

8.248.3.5 isAllocated()

```
bool nntrainer::MemoryPool::isAllocated ( ) const  [virtual]
```

Is the memory pool allocated.

Returns

true if the memory is allocated, else false

Reimplemented in nntrainer::CachePool.

Definition at line 336 of file memory_pool.cpp.

8.248.3.6 minMemoryRequirement()

```
size_t nntrainer::MemoryPool::minMemoryRequirement ( )
```

Get the minimum theoretical memory requirement.

Returns

The theoretical memory requirement with this strategy in bytes

Definition at line 149 of file memory_pool.cpp.

8.248.3.7 planLayout()

```
double nntrainer::MemoryPool::planLayout (
            const MemoryPlanner & planner )
```

Plan the layout with memory planner.

Planner the layout with memory planner.

Parameters

planner	The memory planner to be used for finalizing the layout

Returns

The efficiency of the memory layer with the given memory planner

The efficiency of the planner is calculated as the ratio of the theoretical minimum memory requirement divided by the memory requirement given by the memory planner.

planLayout can be called multiple times as this does not perform any allocation but rather just plans the layout and stores the layout. Subsequent call to this function will overwrite any existing layout.

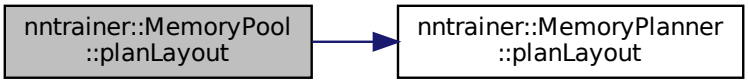
The efficiency of the planner is calculated as the ratio of the theoretical minimum memory requirement divided by the memory requirement given by the memory planner.

planLayout can be called multiple times as this does not perform any allocation but rather just plans the layout and stores the layout. Subsequent call to this function will overwrite any existing layout. mem_pool must be deallocated when planLayout is being called

calculate min_pool_size if not already calculated

Definition at line 67 of file memory_pool.cpp.

Here is the call graph for this function:



8.248.3.8 requestMemory()

```
unsigned int nntrainer::MemoryPool::requestMemory (
            size_t bytes,
            unsigned int start_time,
            unsigned int end_time,
            std::vector< unsigned int > exec_order = std::vector<unsigned int>(),
            TensorLifespan lifespan = TensorLifespan::MAX_LIFESPAN,
            bool is_wgrad = false )  [virtual]
```

Request Memory from memory pool.

Parameters

bytes	The size of the memory requested in bytes
start_time	The start of the validity interval of this memory
end_time	The end of the validity interval of this memory
exec_order	execution orders of this memory
lifespan	lifespan of memory
is_wgrad	check if the tensor is weight gradient

Returns

The token to get the pointer for this memory after allocation

Note

start_time is inclusive, but end_time is exclusive

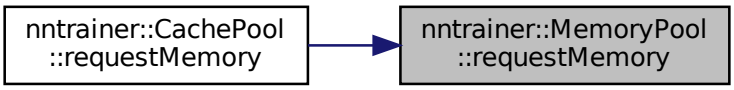
The value of the return token starts from 1.

start_time is inclusive, but end_time is exclusive

invalidate min_pool_size if already there

Definition at line 28 of file memory_pool.cpp.

Here is the caller graph for this function:



8.248.3.9 size()

```
size_t nntrainer::MemoryPool::size ( )
```

Get the maximum real memory requirement.

Returns

The real memory requirement with this strategy in bytes

Definition at line 143 of file memory_pool.cpp.

The documentation for this class was generated from the following files:

- tensor/memory_pool.h
- tensor/memory_pool.cpp

8.249 nntrainer::MemoryRequest Struct Reference

Memory Request data structure clubbing all the requests.

Public Member Functions

- MemoryRequest (size_t s, const std::pair< unsigned int, unsigned int > &valid, unsigned int idx)
  Constructor for the Memory Request.
- MemoryRequest (size_t s, const std::pair< unsigned int, unsigned int > &valid, unsigned int idx)
  Constructor for the Memory Request.
- MemoryRequest (size_t s, const std::pair< unsigned int, unsigned int > &valid, unsigned int idx)
  Constructor for the Memory Request.

Public Attributes

- unsigned int start
- unsigned int end
- unsigned int loc
- size_t size
- size_t offset

8.249.1 Detailed Description

Memory Request data structure clubbing all the requests.

Definition at line 28 of file optimized_v1_planner.cpp.

8.249.2 Constructor & Destructor Documentation

8.249.2.1 MemoryRequest() [1/3]

```
nntrainer::MemoryRequest::MemoryRequest (
            size_t s,
            const std::pair< unsigned int, unsigned int > & valid,
            unsigned int idx )  [inline]
```

Constructor for the Memory Request.

Definition at line 39 of file optimized_v1_planner.cpp.

8.249.2.2 MemoryRequest() [2/3]

```
nntrainer::MemoryRequest::MemoryRequest (
            size_t s,
            const std::pair< unsigned int, unsigned int > & valid,
            unsigned int idx )  [inline]
```

Constructor for the Memory Request.

Definition at line 41 of file optimized_v2_planner.cpp.

8.249.2.3 MemoryRequest() [3/3]

```
nntrainer::MemoryRequest::MemoryRequest (
            size_t s,
            const std::pair< unsigned int, unsigned int > & valid,
            unsigned int idx )  [inline]
```

Constructor for the Memory Request.

Definition at line 39 of file optimized_v3_planner.cpp.

8.249.3 Member Data Documentation

8.249.3.1 end

```
unsigned int nntrainer::MemoryRequest::end
```

end of the validity (exclusive)

Definition at line 30 of file optimized_v1_planner.cpp.

8.249.3.2 loc

```
unsigned int nntrainer::MemoryRequest::loc
```

index/location of the this request

Definition at line 31 of file optimized_v1_planner.cpp.

8.249.3.3 offset

```
size_t nntrainer::MemoryRequest::offset
```

offset for this request

Definition at line 33 of file optimized_v1_planner.cpp.

8.249.3.4 size

```
size_t nntrainer::MemoryRequest::size
```

size of the request

Definition at line 32 of file optimized_v1_planner.cpp.

8.249.3.5 start

```
unsigned int nntrainer::MemoryRequest::start
```

start of the validity (inclusive)

Definition at line 29 of file optimized_v1_planner.cpp.

The documentation for this struct was generated from the following files:

- tensor/optimized_v1_planner.cpp
- tensor/optimized_v2_planner.cpp
- tensor/optimized_v3_planner.cpp

8.250 tflite::MetadataBuilder Struct Reference

Public Types

- typedef Metadata **Table**

Public Member Functions

- void **add_name** (flatbuffers::Offset< flatbuffers::String > name)
- void **add_buffer** (uint32_t buffer)
- **MetadataBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- MetadataBuilder & **operator=** (const MetadataBuilder &)
- flatbuffers::Offset< Metadata > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.250.1 Detailed Description

Definition at line 8242 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.251 tflite::MirrorPadOptionsBuilder Struct Reference

Public Types

- typedef MirrorPadOptions **Table**

Public Member Functions

- void **add_mode** (tflite::MirrorPadMode mode)
- **MirrorPadOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- MirrorPadOptionsBuilder & **operator=** (const MirrorPadOptionsBuilder &)
- flatbuffers::Offset< MirrorPadOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.251.1 Detailed Description

Definition at line 6379 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.252 tflite::ModelBuilder Struct Reference

Public Types

- typedef Model **Table**

Public Member Functions

- void **add_version** (uint32_t version)
- void **add_operator_codes** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::OperatorCode >>> operator_codes)
- void **add_subgraphs** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::SubGraph >>> subgraphs)
- void **add_description** (flatbuffers::Offset< flatbuffers::String > description)
- void **add_buffers** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::Buffer >>> buffers)
- void **add_metadata_buffer** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> metadata_buffer)
- void **add_metadata** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::Metadata >>> metadata)
- void **add_signature_defs** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::SignatureDef >>> signature_defs)
- **ModelBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ModelBuilder & **operator=** (const ModelBuilder &)
- flatbuffers::Offset< Model > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.252.1 Detailed Description

Definition at line 8507 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

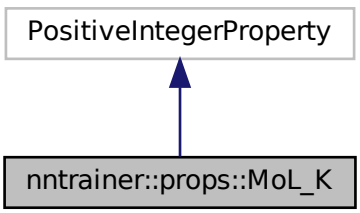
- compiler/tf_schema_generated.h

8.253 nntrainer::props::MoL_K Class Reference

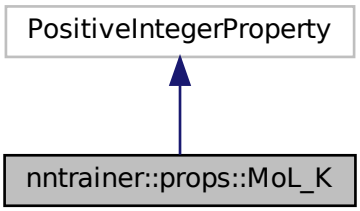
K property, K is the size of the three projections in MoL attention.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::MoL_K:



Collaboration diagram for nntrainer::props::MoL_K:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key = "MoL_K"

8.253.1 Detailed Description

K property, K is the size of the three projections in MoL attention.

Definition at line 1163 of file common_properties.h.

8.253.2 Member Typedef Documentation

8.253.2.1 prop_tag

```
using nntrainer::props::MoL_K::prop_tag = uint_prop_tag
```

property type

Definition at line 1166 of file common_properties.h.

8.253.3 Member Data Documentation

8.253.3.1 key

```
constexpr const char* nntrainer::props::MoL_K::key = "MoL_K"  [static], [constexpr]
```

unique key to access

Definition at line 1165 of file common_properties.h.

The documentation for this class was generated from the following file:

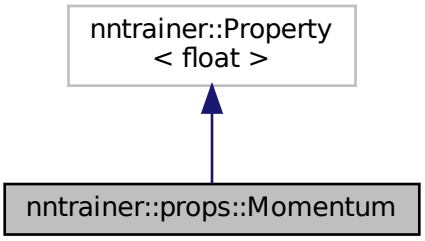
- layers/common_properties.h

8.254 nntrainer::props::Momentum Class Reference

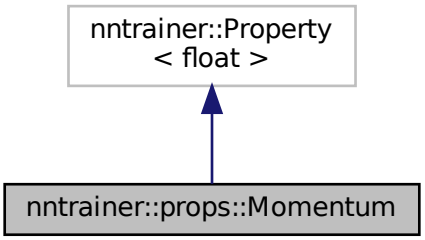
Momentum property, moving average in batch normalization layer.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::Momentum:



Collaboration diagram for nntrainer::props::Momentum:



Public Types

- using prop_tag = float_prop_tag

Public Member Functions

- Momentum (float value=0.99)

  Construct a new Momentum object with a default value 0.99.
- bool isValid (const float &value) const override

  Momentum validator.

Static Public Attributes

- static constexpr const char * key = "momentum"

8.254.1 Detailed Description

Momentum property, moving average in batch normalization layer.

Definition at line 231 of file common_properties.h.

8.254.2 Member Typedef Documentation

8.254.2.1 prop_tag

```
using nntrainer::props::Momentum::prop_tag = float_prop_tag
```

property type

Definition at line 240 of file common_properties.h.

8.254.3 Constructor & Destructor Documentation

8.254.3.1 Momentum()

```
nntrainer::props::Momentum::Momentum (
            float value = 0.99 )
```

Construct a new Momentum object with a default value 0.99.

Definition at line 93 of file common_properties.cpp.

8.254.4 Member Function Documentation

8.254.4.1 isValid()

```
bool nntrainer::props::Momentum::isValid (
            const float & value ) const  [override]
```

Momentum validator.

Parameters

value	float to validate

Return values

true	if it is greater than 0.0 and smaller than 1.0
false	if it is samller or equal than 0.0 or greater or equal than 1.0

Definition at line 95 of file common_properties.cpp.

8.254.5 Member Data Documentation

8.254.5.1 key

```
constexpr const char* nntrainer::props::Momentum::key = "momentum"  [static], [constexpr]
```

unique key to access

Definition at line 239 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.255 tflite::MulOptionsBuilder Struct Reference

Public Types

- typedef MulOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- **MulOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- MulOptionsBuilder & **operator=** (const MulOptionsBuilder &)
- flatbuffers::Offset< MulOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.255.1 Detailed Description

Definition at line 3563 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

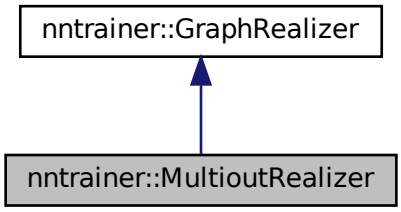
- compiler/tf_schema_generated.h

8.256 nntrainer::MultioutRealizer Class Reference

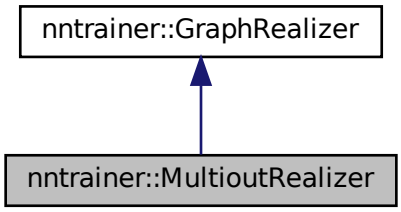
Add multiout layer when a certain input is referenced multiple times.

`#include <multiout_realizer.h>`

Inheritance diagram for nntrainer::MultioutRealizer:



Collaboration diagram for nntrainer::MultioutRealizer:



Public Member Functions

- ~MultioutRealizer ()

  Destroy the Graph Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  graph realizer creates a new graph based on the reference

8.256.1 Detailed Description

Add multiout layer when a certain input is referenced multiple times.

Note

after multiout realizer, it is guaranteed that input_layer only refers to a single connection

Definition at line 28 of file multiout_realizer.h.

8.256.2 Constructor & Destructor Documentation

8.256.2.1 ~MultioutRealizer()

```
nntrainer::MultioutRealizer::~MultioutRealizer ( )
```

Destroy the Graph Realizer object.

Definition at line 26 of file multiout_realizer.cpp.

8.256.3 Member Function Documentation

8.256.3.1 realize()

```
GraphRepresentation nntrainer::MultioutRealizer::realize (
            const GraphRepresentation & reference )  [override], [virtual]
```

graph realizer creates a new graph based on the reference

1. build frequency map and connection names
2. for each connection names, if a connection is referenced multiple times, create multioutput node and remap to multi output node index

< original id

< created node

Note

freq < 1 should never happen as the map entry is not created. but if it happens multiout realizer is not interested in checking if it is a dangled or actually an output. So there is no assurance done at this point. Some other class must check if the given graph is formed in a correct way.

{connection_name}/generated_out_{index}

1. insert multiout_nodes close to the original node to make the realization more sensible

Implements nntrainer::GraphRealizer.

Definition at line 29 of file multiout_realizer.cpp.

The documentation for this class was generated from the following files:

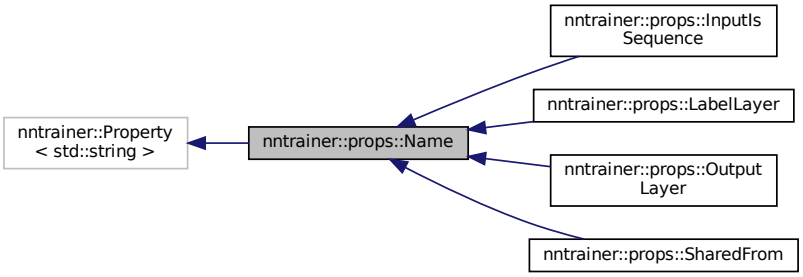
- compiler/multiout_realizer.h
- compiler/multiout_realizer.cpp

8.257 nntrainer::props::Name Class Reference

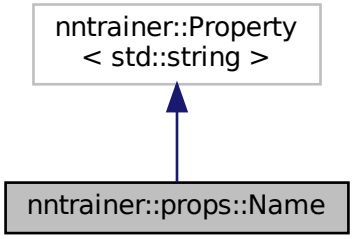
Name property, name is an identifier of an object.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Name:



Collaboration diagram for nntrainer::props::Name:



Public Types

- using prop_tag = str_prop_tag

Public Member Functions

- Name ()

  Construct a new Name object without a default value.
- Name (const std::string &value)

  Construct a new Name object with a default value.
- void set (const std::string &value) override

  Name setter.
- bool isValid (const std::string &v) const override

  name validator

Static Public Attributes

- static constexpr const char * key = "name"

8.257.1 Detailed Description

Name property, name is an identifier of an object.

Definition at line 49 of file common_properties.h.

8.257.2 Member Typedef Documentation

8.257.2.1 prop_tag

```
using nntrainer::props::Name::prop_tag = str_prop_tag
```

property type

Definition at line 65 of file common_properties.h.

8.257.3 Constructor & Destructor Documentation

8.257.3.1 Name() [1/2]

```
nntrainer::props::Name::Name ( )
```

Construct a new Name object without a default value.

Definition at line 30 of file common_properties.cpp.

8.257.3.2 Name() [2/2]

```
nntrainer::props::Name::Name (
            const std::string & value )
```

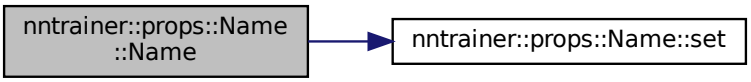
Construct a new Name object with a default value.

Parameters

value	value to contrusct the property

Definition at line 32 of file common_properties.cpp.

Here is the call graph for this function:



8.257.4 Member Function Documentation

8.257.4.1 isValid()

```
bool nntrainer::props::Name::isValid (
            const std::string & v ) const  [override]
```

name validator

Parameters

v	string to validate

Return values

true	if it contains alphanumeric and/or '-', '_', '/'
false	if it is empty or contains non-valid character

Definition at line 44 of file common_properties.cpp.

8.257.4.2 set()

```
void nntrainer::props::Name::set (
            const std::string & value )  [override]
```

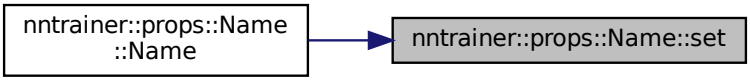
Name setter.

Parameters

value	value to set

Definition at line 34 of file common_properties.cpp.

Here is the caller graph for this function:



8.257.5 Member Data Documentation

8.257.5.1 key

```
constexpr const char* nntrainer::props::Name::key = "name"  [static], [constexpr]
```

unique key to access

Definition at line 64 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.258 tflite::NegOptionsBuilder Struct Reference

Public Types

- typedef NegOptions **Table**

Public Member Functions

- **NegOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- NegOptionsBuilder & **operator=** (const NegOptionsBuilder &)
- flatbuffers::Offset< NegOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.258.1 Detailed Description

Definition at line 5366 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.259 tflite::NonMaxSuppressionV4OptionsBuilder Struct Reference

Public Types

- typedef NonMaxSuppressionV4Options **Table**

Public Member Functions

- **NonMaxSuppressionV4OptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- NonMaxSuppressionV4OptionsBuilder & **operator=** (const NonMaxSuppressionV4OptionsBuilder &)
- flatbuffers::Offset< NonMaxSuppressionV4Options > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.259.1 Detailed Description

Definition at line 6822 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.260 tflite::NonMaxSuppressionV5OptionsBuilder Struct Reference

Public Types

- typedef NonMaxSuppressionV5Options **Table**

Public Member Functions

- **NonMaxSuppressionV5OptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- NonMaxSuppressionV5OptionsBuilder & **operator=** (const NonMaxSuppressionV5OptionsBuilder &)
- flatbuffers::Offset< NonMaxSuppressionV5Options > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.260.1 Detailed Description

Definition at line 6852 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

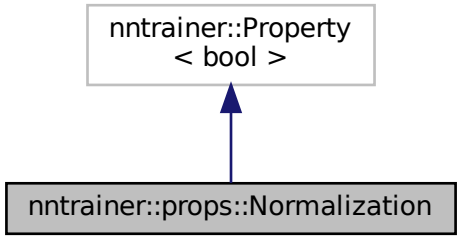
- compiler/tf_schema_generated.h

8.261 nntrainer::props::Normalization Class Reference

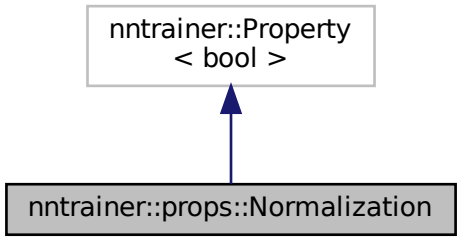
Normalization property, normalize the input to be in range [0, 1] if true.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Normalization:



Collaboration diagram for nntrainer::props::Normalization:



Public Types

- using **prop_tag** = bool_prop_tag

Public Member Functions

- Normalization (bool value=false)

  Construct a new Normalization object.

Static Public Attributes

- static constexpr const char * **key** = "normalization"

8.261.1 Detailed Description

Normalization property, normalize the input to be in range [0, 1] if true.

Definition at line 146 of file common_properties.h.

8.261.2 Constructor & Destructor Documentation

8.261.2.1 Normalization()

```
nntrainer::props::Normalization::Normalization (
            bool value = false )
```

Construct a new Normalization object.

Definition at line 49 of file common_properties.cpp.

The documentation for this class was generated from the following files:

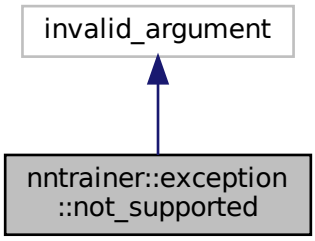
- layers/common_properties.h
- layers/common_properties.cpp

8.262 nntrainer::exception::not_supported Struct Reference

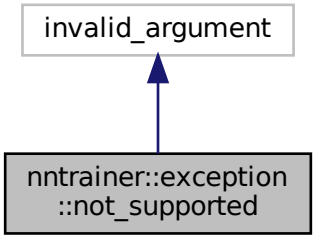
derived class of invalid argument to represent specific functionality not supported

`#include <nntrainer_error.h>`

Inheritance diagram for nntrainer::exception::not_supported:



Collaboration diagram for nntrainer::exception::not_supported:



8.262.1 Detailed Description

derived class of invalid argument to represent specific functionality not supported

Note

this could be either intended or not yet implemented

Definition at line 136 of file nntrainer_error.h.

The documentation for this struct was generated from the following file:

- nntrainer_error.h

8.263 tflite::NotEqualOptionsBuilder Struct Reference

Public Types

- typedef NotEqualOptions **Table**

Public Member Functions

- **NotEqualOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- NotEqualOptionsBuilder & **operator=** (const NotEqualOptionsBuilder &)
- flatbuffers::Offset< NotEqualOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.263.1 Detailed Description

Definition at line 5620 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

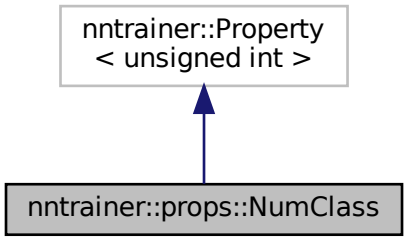
- compiler/tf_schema_generated.h

8.264 nntrainer::props::NumClass Class Reference

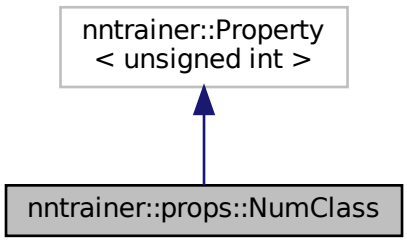
Number of class.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::NumClass:



Collaboration diagram for nntrainer::props::NumClass:



Public Types

- using prop_tag = uint_prop_tag

Public Member Functions

- bool isValid (const unsigned int &v) const override
  int>::isValid(const unsigned int &v);

Static Public Attributes

- static constexpr const char * key = "num_class"

8.264.1 Detailed Description

Number of class.

Todo deprecate this

Definition at line 703 of file common_properties.h.

8.264.2 Member Typedef Documentation

8.264.2.1 prop_tag

```
using nntrainer::props::NumClass::prop_tag = uint_prop_tag
```

property type

Definition at line 705 of file common_properties.h.

8.264.3 Member Function Documentation

8.264.3.1 isValid()

```
bool nntrainer::props::NumClass::isValid (
            const unsigned int & v ) const  [override]
```

int>::isValid(const unsigned int &v);

int>::isValid(const unsigned int &v);

Definition at line 83 of file common_properties.cpp.

8.264.4 Member Data Documentation

8.264.4.1 key

```
constexpr const char* nntrainer::props::NumClass::key = "num_class"  [static], [constexpr]
```

unique key to access

Definition at line 706 of file common_properties.h.

The documentation for this class was generated from the following files:

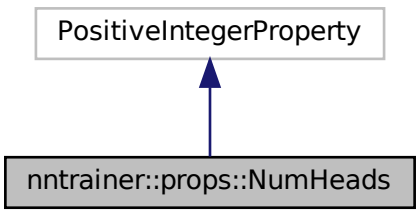
- layers/common_properties.h
- layers/common_properties.cpp

8.265 nntrainer::props::NumHeads Class Reference

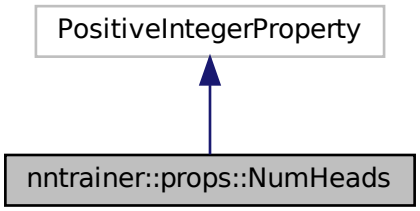
NumHeads property, NumHeads is number of head in multi head attention.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::NumHeads:



Collaboration diagram for nntrainer::props::NumHeads:



Public Types

- using prop_tag = uint_prop_tag

Public Member Functions

- NumHeads (unsigned int value=1)

  Construct a new NumHeads object with default value 1.

Static Public Attributes

- static constexpr const char * key = "num_heads"

8.265.1 Detailed Description

NumHeads property, NumHeads is number of head in multi head attention.

Definition at line 1173 of file common_properties.h.

8.265.2 Member Typedef Documentation

8.265.2.1 prop_tag

```
using nntrainer::props::NumHeads::prop_tag = uint_prop_tag
```

property type

Definition at line 1181 of file common_properties.h.

8.265.3 Constructor & Destructor Documentation

8.265.3.1 NumHeads()

```
nntrainer::props::NumHeads::NumHeads (
            unsigned int value = 1 )
```

Construct a new NumHeads object with default value 1.

Definition at line 339 of file common_properties.cpp.

8.265.4 Member Data Documentation

8.265.4.1 key

```
constexpr const char* nntrainer::props::NumHeads::key = "num_heads"  [static], [constexpr]
```

unique key to access

Definition at line 1180 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.266 tflite::OneHotOptionsBuilder Struct Reference

Public Types

- typedef OneHotOptions **Table**

Public Member Functions

- void **add_axis** (int32_t axis)
- **OneHotOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- OneHotOptionsBuilder & **operator=** (const OneHotOptionsBuilder &)
- flatbuffers::Offset< OneHotOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.266.1 Detailed Description

Definition at line 5913 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.267 Op Class Reference

Class for Optimizer context.

8.267.1 Detailed Description

Class for Optimizer context.

for all optimizers

This provides for the optimizer execution.

Definition at line 23 of file optimizer_context.h.

The documentation for this class was generated from the following file:

- optimizers/optimizer_context.h

8.268 tflite::OperatorBuilder Struct Reference

Public Types

- typedef Operator **Table**

Public Member Functions

- void **add_opcode_index** (uint32_t opcode_index)
- void **add_inputs** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> inputs)
- void **add_outputs** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> outputs)
- void **add_builtin_options_type** (tflite::BuiltinOptions builtin_options_type)
- void **add_builtin_options** (flatbuffers::Offset< void > builtin_options)
- void **add_custom_options** (flatbuffers::Offset< flatbuffers::Vector< uint8_t >> custom_options)
- void **add_custom_options_format** (tflite::CustomOptionsFormat custom_options_format)
- void **add_mutating_variable_inputs** (flatbuffers::Offset< flatbuffers::Vector< uint8_t >> mutating_↩ variable_inputs)
- void **add_intermediates** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> intermediates)
- **OperatorBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- OperatorBuilder & **operator=** (const OperatorBuilder &)
- flatbuffers::Offset< Operator > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.268.1 Detailed Description

Definition at line 7962 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.269 tflite::OperatorCodeBuilder Struct Reference

Public Types

- typedef OperatorCode **Table**

Public Member Functions

- void **add_deprecated_builtin_code** (int8_t deprecated_builtin_code)
- void **add_custom_code** (flatbuffers::Offset< flatbuffers::String > custom_code)
- void **add_version** (int32_t version)
- void **add_builtin_code** (tflite::BuiltinOperator builtin_code)
- **OperatorCodeBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- OperatorCodeBuilder & **operator=** (const OperatorCodeBuilder &)
- flatbuffers::Offset< OperatorCode > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.269.1 Detailed Description

Definition at line 7129 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

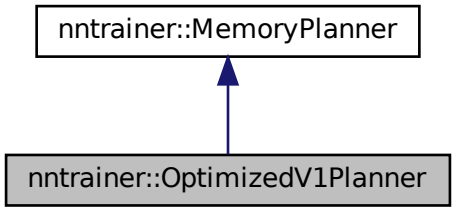
- compiler/tf_schema_generated.h

8.270 nntrainer::OptimizedV1Planner Class Reference

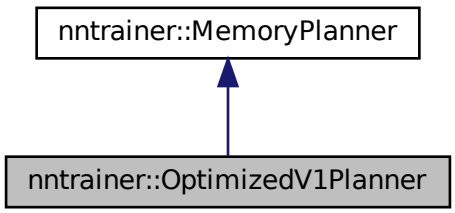
Optimized V1 Memory Planner provides the optimized plan for memory layout.

`#include <optimized_v1_planner.h>`

Inheritance diagram for nntrainer::OptimizedV1Planner:



Collaboration diagram for nntrainer::OptimizedV1Planner:



Public Member Functions

- OptimizedV1Planner ()=default

  OptimizedV1Planner destructor.
- size_t planLayout (const std::vector< size_t > &memory_size, const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, std::vector< size_t > &memory_offset, std::vector< bool > &memory_is_wgrad, size_t n_wgrad=0) const
- const std::string & getType () const

  Get type of the planner.

Static Public Attributes

- static const std::string **type** = "optimized_v1_planner"

8.270.1 Detailed Description

Optimized V1 Memory Planner provides the optimized plan for memory layout.

optimized planner performs sharing of overlapping memory sharing upto certain extent

Definition at line 44 of file optimized_v1_planner.h.

8.270.2 Constructor & Destructor Documentation

8.270.2.1 OptimizedV1Planner()

```
nntrainer::OptimizedV1Planner::OptimizedV1Planner ( )  [default]
```

OptimizedV1Planner destructor.

8.270.3 Member Function Documentation

8.270.3.1 getType()

```
const std::string& nntrainer::OptimizedV1Planner::getType ( ) const  [inline], [virtual]
```

Get type of the planner.

Returns

The type of the planner

Implements nntrainer::MemoryPlanner.

Definition at line 70 of file optimized_v1_planner.h.

8.270.3.2 planLayout()

```
size_t nntrainer::OptimizedV1Planner::planLayout (
            const std::vector< size_t > & memory_size,
            const std::vector< std::pair< unsigned int, unsigned int >> & memory_validity,
            std::vector< size_t > & memory_offset,
            std::vector< bool > & memory_is_wgrad,
            size_t n_wgrad = 0 ) const  [virtual]
```

The optimized v1 memory planner assigns memory to the requests whose validity starts first. The requested memories are sorted based on the ascending order of the start timestamps, and descending order using the end timestamps. The sorted memories are given increasing offset based on the memory size. At the end of each timestamp, invalid memories are freed, and offset updated for reuse. This planner allocates overlapping memory for all the required memories. create memory requests structure array for easier management

sort the memory requests with ascending order of start time first, and then end time

TODO: try this

all the memories in use sorted by their assigned offset and size

iterate over the sorted requests and start allocation of the requests

remove expired memories and update offset

if there exists an expired memory with same size (not at the edge), reuse it

TODO: reuse if memory size not exactly match

assign offset to the new request and push to queue

Implements nntrainer::MemoryPlanner.

Definition at line 128 of file optimized_v1_planner.cpp.

The documentation for this class was generated from the following files:

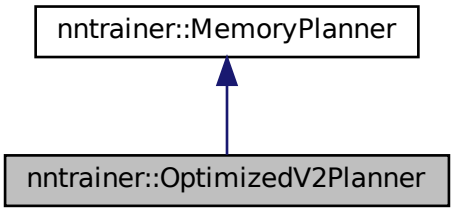
- tensor/optimized_v1_planner.h
- tensor/optimized_v1_planner.cpp

8.271 nntrainer::OptimizedV2Planner Class Reference

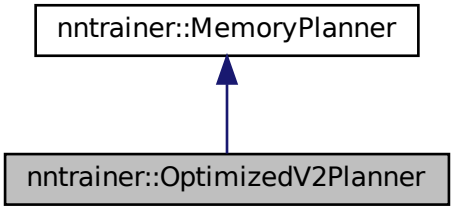
Optimized V2 Memory Planner provides the optimized plan for memory layout.

```
#include <optimized_v2_planner.h>
```

Inheritance diagram for nntrainer::OptimizedV2Planner:



Collaboration diagram for nntrainer::OptimizedV2Planner:



Public Member Functions

- OptimizedV2Planner ()=default

  OptimizedV1Planner destructor.
- size_t planLayout (const std::vector< size_t > &memory_size, const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, std::vector< size_t > &memory_offset, std::vector< bool > &memory_is_wgrad, size_t n_wgrad=0) const
- const std::string & getType () const

  Get type of the planner.

Static Public Attributes

- static const std::string **type** = "optimized_v2_planner"

8.271.1 Detailed Description

Optimized V2 Memory Planner provides the optimized plan for memory layout.

optimized planner performs sharing of overlapping memory sharing upto certain extent

Definition at line 45 of file optimized_v2_planner.h.

8.271.2 Constructor & Destructor Documentation

8.271.2.1 OptimizedV2Planner()

```
nntrainer::OptimizedV2Planner::OptimizedV2Planner ( )  [default]
```

OptimizedV1Planner destructor.

8.271.3 Member Function Documentation

8.271.3.1 getType()

```
const std::string& nntrainer::OptimizedV2Planner::getType ( ) const  [inline], [virtual]
```

Get type of the planner.

Returns

The type of the planner

Implements nntrainer::MemoryPlanner.

Definition at line 71 of file optimized_v2_planner.h.

8.271.3.2 planLayout()

```
size_t nntrainer::OptimizedV2Planner::planLayout (
            const std::vector< size_t > & memory_size,
            const std::vector< std::pair< unsigned int, unsigned int >> & memory_validity,
            std::vector< size_t > & memory_offset,
            std::vector< bool > & memory_is_wgrad,
            size_t n_wgrad = 0 ) const  [virtual]
```

The optimized v1 memory planner assigns memory to the requests whose validity starts first. The requested memories are sorted based on the ascending order of the start timestamps, and descending order using the end timestamps. The sorted memories are given increasing offset based on the memory size. At the end of each timestamp, invalid memories are freed, and offset updated for reuse. This planner allocates overlapping memory for all the required memories. create memory requests structure array for easier management

sort the memory requests with ascending order of start time first, and then end time

TODO: try this

all the memories in use sorted by their assigned offset and size

iterate over the sorted requests and start allocation of the requests

remove expired memories and update offset

if there exists an expired memory with same size (not at the edge), reuse it

TODO: reuse if memory size not exactly match

assign offset to the new request and push to queue

TODO: We donot need to start from memeory_req. We might find proper offset considering execution order

Implements nntrainer::MemoryPlanner.

Definition at line 141 of file optimized_v2_planner.cpp.

The documentation for this class was generated from the following files:

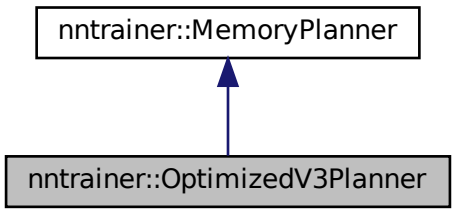
- tensor/optimized_v2_planner.h
- tensor/optimized_v2_planner.cpp

8.272 nntrainer::OptimizedV3Planner Class Reference

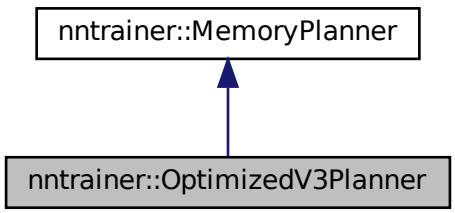
Optimized V3 Memory Planner provides the optimized plan for memory layout.

`#include <optimized_v3_planner.h>`

Inheritance diagram for nntrainer::OptimizedV3Planner:



Collaboration diagram for nntrainer::OptimizedV3Planner:



Public Member Functions

- OptimizedV3Planner ()=default

  OptimizedV3Planner destructor.
- size_t planLayout (const std::vector< size_t > &memory_size, const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, std::vector< size_t > &memory_offset, std::vector< bool > &memory_is_wgrad, size_t n_wgrad=0) const
- const std::string & getType () const

  Get type of the planner.

Static Public Attributes

- static const std::string **type** = "optimized_v3_planner"

8.272.1 Detailed Description

Optimized V3 Memory Planner provides the optimized plan for memory layout.

optimized planner performs sharing of overlapping memory sharing upto certain extent

Definition at line 31 of file optimized_v3_planner.h.

8.272.2 Constructor & Destructor Documentation

8.272.2.1 OptimizedV3Planner()

```
nntrainer::OptimizedV3Planner::OptimizedV3Planner ( )  [default]
```

OptimizedV3Planner destructor.

8.272.3 Member Function Documentation

8.272.3.1 getType()

```
const std::string& nntrainer::OptimizedV3Planner::getType ( ) const  [inline], [virtual]
```

Get type of the planner.

Returns

The type of the planner

Implements nntrainer::MemoryPlanner.

Definition at line 57 of file optimized_v3_planner.h.

8.272.3.2 planLayout()

```
size_t nntrainer::OptimizedV3Planner::planLayout (
            const std::vector< size_t > & memory_size,
            const std::vector< std::pair< unsigned int, unsigned int >> & memory_validity,
            std::vector< size_t > & memory_offset,
            std::vector< bool > & memory_is_wgrad,
            size_t n_wgrad = 0 ) const  [virtual]
```

The optimized v1 memory planner assigns memory to the requests whose validity starts first. The requested memories are sorted based on the ascending order of the start timestamps, and descending order using the end timestamps. The sorted memories are given increasing offset based on the memory size. At the end of each timestamp, invalid memories are freed, and offset updated for reuse. This planner allocates overlapping memory for all the required memories. create memory requests structure array for easier management

sort the memory requests with ascending order of start time first, and then end time

TODO: try this

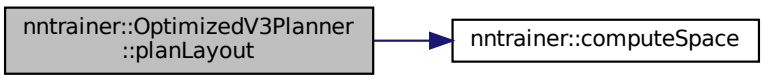
all the memories in use sorted by their assigned offset and size

iterate over the sorted requests and start allocation of the requests

Implements nntrainer::MemoryPlanner.

Definition at line 159 of file optimized_v3_planner.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

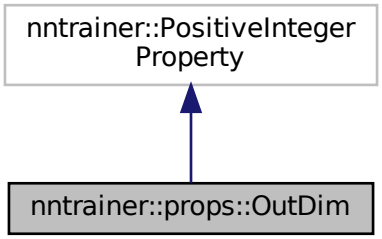
- tensor/optimized_v3_planner.h
- tensor/optimized_v3_planner.cpp

8.273 nntrainer::props::OutDim Class Reference

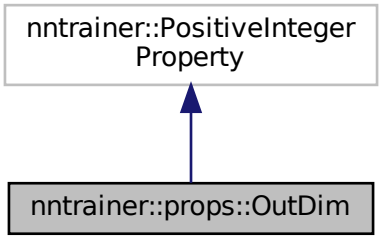
OutDim property, out dim is the size of the vector space in which words will be embedded.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::OutDim:



Collaboration diagram for nntrainer::props::OutDim:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key = "out_dim"

8.273.1 Detailed Description

OutDim property, out dim is the size of the vector space in which words will be embedded.

Definition at line 482 of file common_properties.h.

8.273.2 Member Typedef Documentation

8.273.2.1 prop_tag

```
using nntrainer::props::OutDim::prop_tag = uint_prop_tag
```

property type

Definition at line 485 of file common_properties.h.

8.273.3 Member Data Documentation

8.273.3.1 key

```
constexpr const char* nntrainer::props::OutDim::key = "out_dim"  [static], [constexpr]
```

unique key to access

Definition at line 484 of file common_properties.h.

The documentation for this class was generated from the following file:

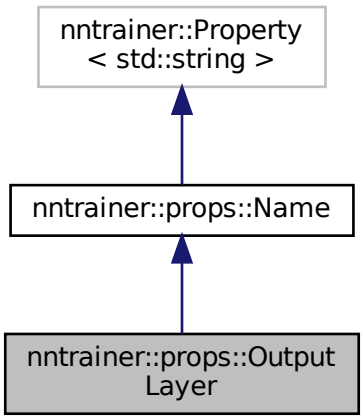
- layers/common_properties.h

8.274 nntrainer::props::OutputLayer Class Reference

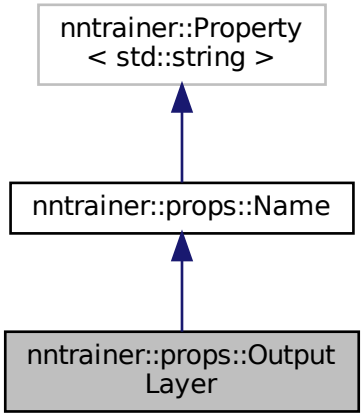
Output Layer name property which saves a single connection (practically, std::vector<InputLayers> is used)

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::OutputLayer:



Collaboration diagram for nntrainer::props::OutputLayer:



Public Types

- using **prop_tag** = str_prop_tag

Public Member Functions

- OutputLayer ()

  Construct a new Output Layer object.
- OutputLayer (const std::string &name)

  Construct a new Output Layer object.

Static Public Attributes

- static constexpr const char * **key** = "output_layers"

8.274.1 Detailed Description

Output Layer name property which saves a single connection (practically, std::vector<InputLayers> is used)

Definition at line 795 of file common_properties.h.

8.274.2 Constructor & Destructor Documentation

8.274.2.1 OutputLayer() [1/2]

```
nntrainer::props::OutputLayer::OutputLayer ( )
```

Construct a new Output Layer object.

Definition at line 283 of file common_properties.cpp.

8.274.2.2 OutputLayer() [2/2]

```
nntrainer::props::OutputLayer::OutputLayer (
            const std::string & name )
```

Construct a new Output Layer object.

Parameters

name	name to set

Definition at line 284 of file common_properties.cpp.

The documentation for this class was generated from the following files:

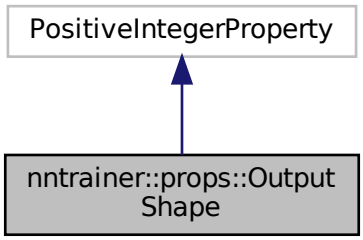
- layers/common_properties.h
- layers/common_properties.cpp

8.275 nntrainer::props::OutputShape Class Reference

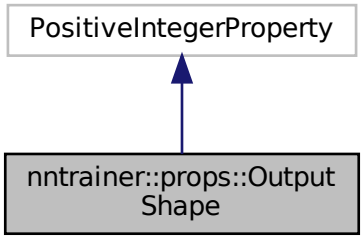
OutputShape property, output shape of multi head attention.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::OutputShape:



Collaboration diagram for nntrainer::props::OutputShape:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key

8.275.1 Detailed Description

OutputShape property, output shape of multi head attention.

Correspond with output_shape of tensorflow

Definition at line 1216 of file common_properties.h.

8.275.2 Member Typedef Documentation

8.275.2.1 prop_tag

```
using nntrainer::props::OutputShape::prop_tag = uint_prop_tag
```

property type

Definition at line 1220 of file common_properties.h.

8.275.3 Member Data Documentation

8.275.3.1 key

```
constexpr const char* nntrainer::props::OutputShape::key  [static], [constexpr]
```

Initial value:

```
=
    "output_shape"
```

unique key to access

Definition at line 1218 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.276 tflite::PackOptionsBuilder Struct Reference

Public Types

- typedef PackOptions **Table**

Public Member Functions

- void **add_values_count** (int32_t values_count)
- void **add_axis** (int32_t axis)
- **PackOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- PackOptionsBuilder & **operator=** (const PackOptionsBuilder &)
- flatbuffers::Offset< PackOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.276.1 Detailed Description

Definition at line 5836 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

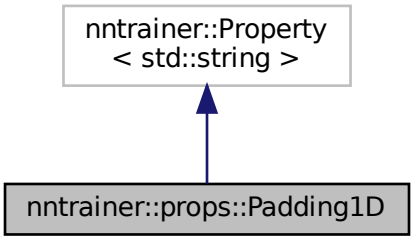
- compiler/tf_schema_generated.h

8.277 nntrainer::props::Padding1D Class Reference

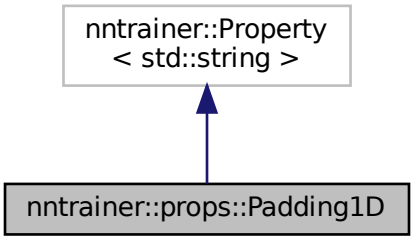
Padding1D property, this is used to calculate padding2D.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Padding1D:



Collaboration diagram for nntrainer::props::Padding1D:



Public Types

- using prop_tag = str_prop_tag

Public Member Functions

- bool isValid (const std::string &v) const override
- std::array< unsigned int, 2 > compute (const TensorDim &input, const unsigned int &kernel, const unsigned int &stride, const unsigned int &dilation)

  compute actual padding1d from the underlying data

Static Public Attributes

- static constexpr const char * key = "padding"

8.277.1 Detailed Description

Padding1D property, this is used to calculate padding2D.

Padding1D is saved as a string. Upon calling Padding1D::compute, returns std::vector<unsigned int> which has computed padding1Ds, below formats are accepted valid

1. "same" (case insensitive literal string)
2. "valid" (case insensitive literal string)

1. "causal" (case insensitive literal string)
2. "padding1d_all", eg) padding=1
3. "padding1d_left, padding1d_right" eg) padding=1,1

Definition at line 441 of file common_properties.h.

8.277.2 Member Typedef Documentation

8.277.2.1 prop_tag

```
using nntrainer::props::Padding1D::prop_tag = str_prop_tag
```

property type

Definition at line 451 of file common_properties.h.

8.277.3 Member Function Documentation

8.277.3.1 compute()

```
std::array< unsigned int, 2 > nntrainer::props::Padding1D::compute (
            const TensorDim & input,
            const unsigned int & kernel,
            const unsigned int & stride,
            const unsigned int & dilation )
```

compute actual padding1d from the underlying data

Parameters

input	input dimension
kernel	kernel dimension
stride	stride

Returns

std::array<unsigned int, 4> list of unsigned padding

padding representation

ceil(input / stride)

case 4, 5: padding has a sequence of unsigned integer

Definition at line 224 of file common_properties.cpp.

8.277.3.2 isValid()

```
bool nntrainer::props::Padding1D::isValid (
            const std::string & v ) const  [override]
```

case 1, 2, 3: padding has string literal

case 4, 5: padding has a sequence of unsigned integer

check if every padding is non-negative integer

case else: false

Definition at line 199 of file common_properties.cpp.

8.277.4 Member Data Documentation

8.277.4.1 key

```
constexpr const char* nntrainer::props::Padding1D::key = "padding"  [static], [constexpr]
```

unique key to access

Definition at line 450 of file common_properties.h.

The documentation for this class was generated from the following files:

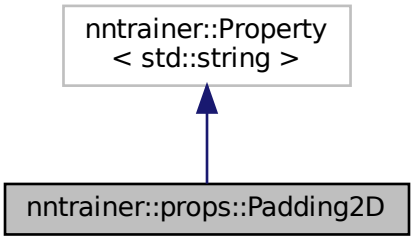
- layers/common_properties.h
- layers/common_properties.cpp

8.278 nntrainer::props::Padding2D Class Reference

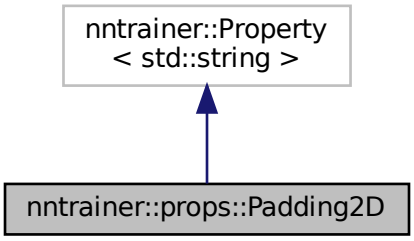
Padding2D property, this is used to calculate padding2D.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Padding2D:



Collaboration diagram for nntrainer::props::Padding2D:



Public Types

- using prop_tag = str_prop_tag

Public Member Functions

- bool isValid (const std::string &v) const override
- std::array< unsigned int, 4 > compute (const TensorDim &input, const TensorDim &kernel, const std::array< unsigned int, 2 > &strides, const std::array< unsigned int, 2 > &dilation)

  compute actual padding2D from the underlying data

Static Public Attributes

- static constexpr const char * key = "padding"

8.278.1 Detailed Description

Padding2D property, this is used to calculate padding2D.

Padding2D is saved as a string. Upon calling Padding2D::compute, returns std::vector<unsigned int> which has computed padding2Ds, below formats are accepted valid

1. "same" (case insensitive literal string)
2. "valid" (case insensitive literal string)
3. "padding2D_all", eg) padding=1
4. "padding2D_height, padding2D_width" eg) padding=1,1
5. "padding2D_top, padding2D_bottom, padding2D_left, padding2D_right" eg) padding=1,1,1,1

Definition at line 403 of file common_properties.h.

8.278.2 Member Typedef Documentation

8.278.2.1 prop_tag

```
using nntrainer::props::Padding2D::prop_tag = str_prop_tag
```

property type

Definition at line 413 of file common_properties.h.

8.278.3 Member Function Documentation

8.278.3.1 compute()

```
std::array< unsigned int, 4 > nntrainer::props::Padding2D::compute (
            const TensorDim & input,
            const TensorDim & kernel,
            const std::array< unsigned int, 2 > & strides,
            const std::array< unsigned int, 2 > & dilation )
```

compute actual padding2D from the underlying data

Parameters

input	input dimension
kernel	kernel dimension
stride	stride

Returns

std::array<unsigned int, 4> list of unsigned padding

padding representation

in the case of same padding, padding is distributed to each side if possible. otherwise pad_all_side / 2 is allocated to top | left and rest are assigned to the other side

ceil(input / stride)

case 3, 4, 5: padding has a sequence of unsigned integer

Definition at line 148 of file common_properties.cpp.

8.278.3.2 isValid()

```
bool nntrainer::props::Padding2D::isValid (
            const std::string & v ) const  [override]
```

case 1, 2: padding has string literal

case 3, 4, 5: padding has a sequence of unsigned integer

check if every padding is non-negative integer

case else: false

Definition at line 122 of file common_properties.cpp.

8.278.4 Member Data Documentation

8.278.4.1 key

```
constexpr const char* nntrainer::props::Padding2D::key = "padding"  [static], [constexpr]
```

unique key to access

Definition at line 412 of file common_properties.h.

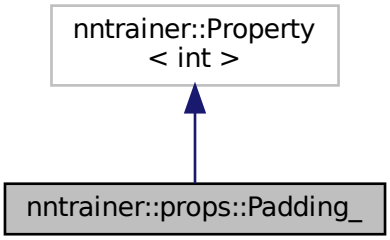
The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

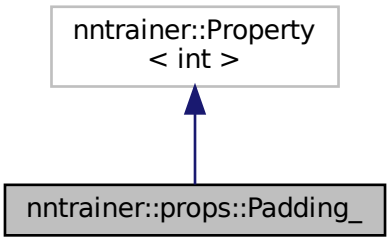
8.279 nntrainer::props::Padding_ Class Reference

unsigned integer property, internally used to parse padding values

Inheritance diagram for nntrainer::props::Padding_:



Collaboration diagram for nntrainer::props::Padding_:



Public Types

- using prop_tag = int_prop_tag

8.279.1 Detailed Description

unsigned integer property, internally used to parse padding values

Definition at line 117 of file common_properties.cpp.

8.279.2 Member Typedef Documentation

8.279.2.1 prop_tag

```
using nntrainer::props::Padding_::prop_tag = int_prop_tag
```

property type

Definition at line 119 of file common_properties.cpp.

The documentation for this class was generated from the following file:

- layers/common_properties.cpp

8.280 tflite::PadOptionsBuilder Struct Reference

Public Types

- typedef PadOptions **Table**

Public Member Functions

- **PadOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- PadOptionsBuilder & **operator=** (const PadOptionsBuilder &)
- flatbuffers::Offset< PadOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.280.1 Detailed Description

Definition at line 4114 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.281 tflite::PadV2OptionsBuilder Struct Reference

Public Types

- typedef PadV2Options **Table**

Public Member Functions

- **PadV2OptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- PadV2OptionsBuilder & **operator=** (const PadV2OptionsBuilder &)
- flatbuffers::Offset< PadV2Options > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.281.1 Detailed Description

Definition at line 4144 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.282 nntrainer::ParallelBatch Class Reference

ParallelBatch class to parallelize along batch direction.

`#include <nntr_threads.h>`

Public Member Functions

- ParallelBatch (unsigned int batch)

  Construct a new ParallelBatch object.
- ParallelBatch (threaded_cb threaded_cb_, unsigned int batch, void *user_data_)

  Construct a new ParallelBatch object.
- ~ParallelBatch ()

  Destroy the ParallelBatch object.
- void run ()

  Run the workders.
- void setCallback (threaded_cb t_cb, void *user_data)

  set the thread callback function
- unsigned int getNumWorkers ()

  return the number of workders

8.282.1 Detailed Description

ParallelBatch class to parallelize along batch direction.

Definition at line 34 of file nntr_threads.h.

8.282.2 Constructor & Destructor Documentation

8.282.2.1 ParallelBatch() [1/2]

```
nntrainer::ParallelBatch::ParallelBatch (
            unsigned int batch )
```

Construct a new ParallelBatch object.

Parameters

unsigned	int total number of batch

Definition at line 18 of file nntr_threads.cpp.

8.282.2.2 ParallelBatch() [2/2]

```
nntrainer::ParallelBatch::ParallelBatch (
            threaded_cb threaded_cb_,
            unsigned int batch,
            void * user_data_ )
```

Construct a new ParallelBatch object.

Parameters

threaded_cb	the function run in thread
unsigned	int total number of batch
void∗	user data for the threaded callback function

Definition at line 24 of file nntr_threads.cpp.

8.282.2.3 ∼ParallelBatch()

```
nntrainer::ParallelBatch::~ParallelBatch ( )
```

Destroy the ParallelBatch object.

Definition at line 31 of file nntr_threads.cpp.

8.282.3 Member Function Documentation

8.282.3.1 getNumWorkers()

```
unsigned int nntrainer::ParallelBatch::getNumWorkers ( )  [inline]
```

return the number of workders

Returns

unsigned int the number of workers

Definition at line 77 of file nntr_threads.h.

8.282.3.2 run()

```
void nntrainer::ParallelBatch::run ( )
```

Run the workders.

Definition at line 33 of file nntr_threads.cpp.

8.282.3.3 setCallback()

```
void nntrainer::ParallelBatch::setCallback (
            threaded_cb t_cb,
            void * user_data )
```

set the thread callback function

Parameters

Threadedcb	the function run in thread
void∗	user data for the threaded callback function

Definition at line 56 of file nntr_threads.cpp.

The documentation for this class was generated from the following files:

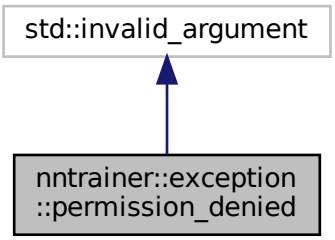
- utils/nntr_threads.h
- utils/nntr_threads.cpp

8.283 nntrainer::exception::permission_denied Struct Reference

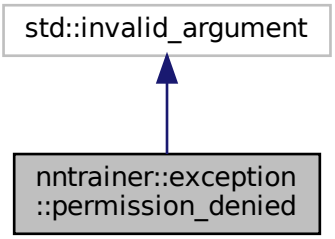
derived class of invalid argument to represent permission is denied

```
#include <nntrainer_error.h>
```

Inheritance diagram for nntrainer::exception::permission_denied:



Collaboration diagram for nntrainer::exception::permission_denied:



8.283.1 Detailed Description

derived class of invalid argument to represent permission is denied

Definition at line 143 of file nntrainer_error.h.

The documentation for this struct was generated from the following file:

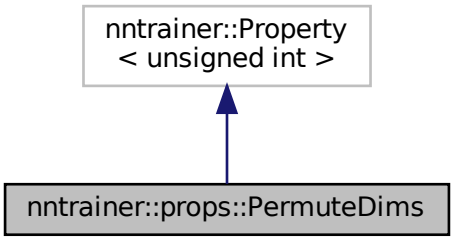
- nntrainer_error.h

8.284 nntrainer::props::PermuteDims Class Reference

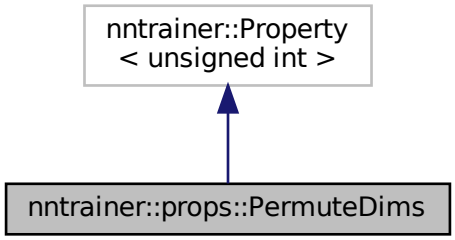
PermuteDims property, direction property describes the axis to be transposed. to be used with array.

`#include <permute_layer.h>`

Inheritance diagram for nntrainer::props::PermuteDims:



Collaboration diagram for nntrainer::props::PermuteDims:



Public Types

- using prop_tag = uint_prop_tag

Public Member Functions

- bool isValid (const unsigned int &) const override
  check if given value is valid

Static Public Attributes

- static constexpr const char * key = "direction"

8.284.1 Detailed Description

PermuteDims property, direction property describes the axis to be transposed. to be used with array.

Definition at line 31 of file permute_layer.h.

8.284.2 Member Typedef Documentation

8.284.2.1 prop_tag

```
using nntrainer::props::PermuteDims::prop_tag = uint_prop_tag
```

property type

Definition at line 34 of file permute_layer.h.

8.284.3 Member Function Documentation

8.284.3.1 isValid()

```
bool nntrainer::props::PermuteDims::isValid (
            const unsigned int & value ) const  [override]
```

check if given value is valid

Returns

true if valid

false if not valid

Definition at line 28 of file permute_layer.cpp.

8.284.4 Member Data Documentation

8.284.4.1 key

```
constexpr const char* nntrainer::props::PermuteDims::key = "direction"  [static], [constexpr]
```

unique key to access

Definition at line 33 of file permute_layer.h.

The documentation for this class was generated from the following files:

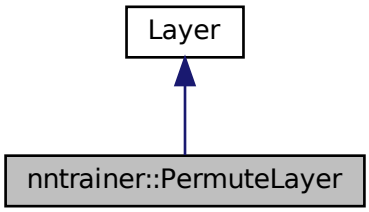
- layers/permute_layer.h
- layers/permute_layer.cpp

8.285 nntrainer::PermuteLayer Class Reference

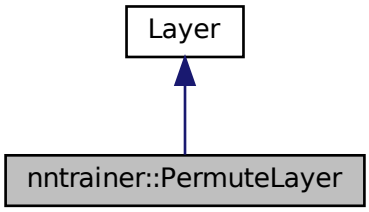
Permute layer to transpose a tensor.

`#include <permute_layer.h>`

Inheritance diagram for nntrainer::PermuteLayer:



Collaboration diagram for nntrainer::PermuteLayer:



Public Member Functions

- PermuteLayer ()
  Constructor of Permute Layer.
- ~PermuteLayer ()=default
  Destructor of Permute Layer.
- PermuteLayer (PermuteLayer &&rhs) noexcept=default
  Move constructor.
- PermuteLayer & operator= (PermuteLayer &&rhs)=default
  Move assignment operator.
- void finalize (InitLayerContext &context) override
- void forwarding (RunLayerContext &context, bool training) override
- void calcDerivative (RunLayerContext &context) override
- void exportTo (Exporter &exporter, const ml::train::ExportMethods &method) const override
- const std::string getType () const override
- bool supportBackwarding () const override
- void setProperty (const std::vector< std::string > &values) override

Static Public Attributes

- static const std::string **type** = "permute"

8.285.1 Detailed Description

Permute layer to transpose a tensor.

Definition at line 50 of file permute_layer.h.

8.285.2 Constructor & Destructor Documentation

8.285.2.1 PermuteLayer() [1/2]

```
nntrainer::PermuteLayer::PermuteLayer ( )  [inline]
```

Constructor of Permute Layer.

Parameters

direction	direction to permute

Definition at line 56 of file permute_layer.h.

8.285.2.2 PermuteLayer() [2/2]

```
nntrainer::PermuteLayer::PermuteLayer (
            PermuteLayer && rhs )  [default], [noexcept]
```

Move constructor.

Parameters

in	*PermuteLayer*	&&

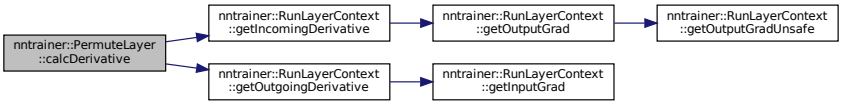
8.285.3 Member Function Documentation

8.285.3.1 calcDerivative()

```
void nntrainer::PermuteLayer::calcDerivative (
            RunLayerContext & context )  [override]
```

Definition at line 77 of file permute_layer.cpp.

Here is the call graph for this function:

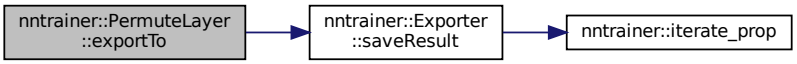


8.285.3.2 exportTo()

```
void nntrainer::PermuteLayer::exportTo (
            Exporter & exporter,
            const ml::train::ExportMethods & method ) const  [override]
```

Definition at line 84 of file permute_layer.cpp.

Here is the call graph for this function:



8.285.3.3 finalize()

```
void nntrainer::PermuteLayer::finalize (
            InitLayerContext & context )  [override]
```

< check if transpose contains all axis

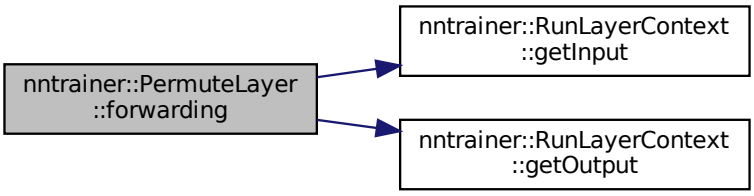
Definition at line 46 of file permute_layer.cpp.

8.285.3.4 forwarding()

```
void nntrainer::PermuteLayer::forwarding (
            RunLayerContext & context,
            bool training ) [override]
```

Definition at line 70 of file permute_layer.cpp.

Here is the call graph for this function:



8.285.3.5 getType()

```
const std::string nntrainer::PermuteLayer::getType ( ) const [inline], [override]
```

Definition at line 100 of file permute_layer.h.

8.285.3.6 operator=()

```
PermuteLayer& nntrainer::PermuteLayer::operator= (
            PermuteLayer && rhs ) [default]
```

Move assignment operator.

Parameters

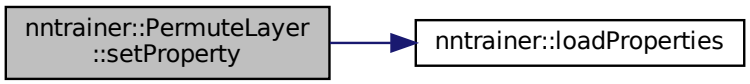
in	*rhs*	PermuteLayer to be moved.

8.285.3.7 setProperty()

```
void nntrainer::PermuteLayer::setProperty (
            const std::vector< std::string > & values ) [override]
```

Definition at line 89 of file permute_layer.cpp.

Here is the call graph for this function:



8.285.3.8 supportBackwarding()

```
bool nntrainer::PermuteLayer::supportBackwarding ( ) const  [inline], [override]
```

Definition at line 105 of file permute_layer.h.

The documentation for this class was generated from the following files:

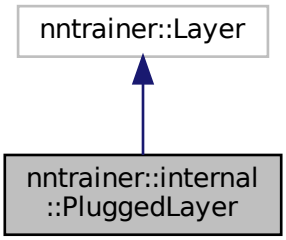
- layers/permute_layer.h
- layers/permute_layer.cpp

8.286 nntrainer::internal::PluggedLayer Class Reference

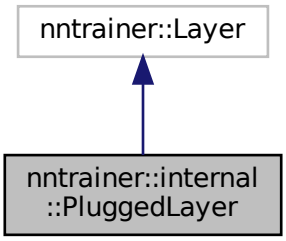
PluggedLayer to wrap a layer from shared object file.

```
#include <plugged_layer.h>
```

Inheritance diagram for nntrainer::internal::PluggedLayer:



Collaboration diagram for nntrainer::internal::PluggedLayer:



Public Member Functions

- PluggedLayer (const nntrainer::LayerPluggable *pluggable)
  Construct a new Plugged Layer object.
- ~PluggedLayer () override
  Destroy the Plugged Layer object.
- PluggedLayer (PluggedLayer &&rhs) noexcept=default
  Move Contruct Plugged Layer object.
- PluggedLayer & operator= (PluggedLayer &&rhs)=default
  Move assign Plugged Layer Object.
- const std::string getType () const override
- void finalize (InitLayerContext &context) override
- void forwarding (RunLayerContext &context, bool training) override
- void calcDerivative (RunLayerContext &context) override
- void calcGradient (RunLayerContext &context) override
- void setProperty (const std::vector< std::string > &values) override
- void exportTo (Exporter &exporter, const ml::train::ExportMethods &method) const override
- void setBatch (RunLayerContext &context, unsigned int batch) override
- bool supportInPlace () const override
- bool requireLabel () const override
- bool supportBackwarding () const override

8.286.1 Detailed Description

PluggedLayer to wrap a layer from shared object file.

Definition at line 28 of file plugged_layer.h.

8.286.2 Constructor & Destructor Documentation

8.286.2.1 PluggedLayer() [1/2]

```
nntrainer::internal::PluggedLayer::PluggedLayer (
            const nntrainer::LayerPluggable * pluggable )  [inline]
```

Construct a new Plugged Layer object.

Parameters

pluggable	LayerPluggable structure from the symbol

Definition at line 35 of file plugged_layer.h.

8.286.2.2 ~PluggedLayer()

```
nntrainer::internal::PluggedLayer::~PluggedLayer ( )  [inline], [override]
```

Destroy the Plugged Layer object.

Definition at line 46 of file plugged_layer.h.

8.286.2.3 PluggedLayer() [2/2]

```
nntrainer::internal::PluggedLayer::PluggedLayer (
            PluggedLayer && rhs )  [default], [noexcept]
```

Move Contruct Plugged Layer object.

Parameters

rhs	layer to move

8.286.3 Member Function Documentation

8.286.3.1 calcDerivative()

```
void nntrainer::internal::PluggedLayer::calcDerivative (
            RunLayerContext & context )  [inline], [override]
```

Definition at line 85 of file plugged_layer.h.

8.286.3.2 calcGradient()

```
void nntrainer::internal::PluggedLayer::calcGradient (
            RunLayerContext & context )  [inline], [override]
```

Definition at line 92 of file plugged_layer.h.

8.286.3.3 exportTo()

```
void nntrainer::internal::PluggedLayer::exportTo (
            Exporter & exporter,
            const ml::train::ExportMethods & method ) const  [inline], [override]
```

Definition at line 108 of file plugged_layer.h.

8.286.3.4 finalize()

```
void nntrainer::internal::PluggedLayer::finalize (
            InitLayerContext & context )  [inline], [override]
```

Definition at line 71 of file plugged_layer.h.

8.286.3.5 forwarding()

```
void nntrainer::internal::PluggedLayer::forwarding (
            RunLayerContext & context,
            bool training )  [inline], [override]
```

Definition at line 78 of file plugged_layer.h.

8.286.3.6 getType()

```
const std::string nntrainer::internal::PluggedLayer::getType ( ) const  [inline], [override]
```

Definition at line 66 of file plugged_layer.h.

8.286.3.7 operator=()

```
PluggedLayer& nntrainer::internal::PluggedLayer::operator= (
            PluggedLayer && rhs )  [default]
```

Move assign Plugged Layer Object.

Parameters

rhs	layer to move

Returns

PluggedLayer& ∗this

8.286.3.8 requireLabel()

```
bool nntrainer::internal::PluggedLayer::requireLabel ( ) const  [inline], [override]
```

Definition at line 128 of file plugged_layer.h.

8.286.3.9 setBatch()

```
void nntrainer::internal::PluggedLayer::setBatch (
            RunLayerContext & context,
            unsigned int batch )  [inline], [override]
```

Definition at line 116 of file plugged_layer.h.

8.286.3.10 setProperty()

```
void nntrainer::internal::PluggedLayer::setProperty (
            const std::vector< std::string > & values )  [inline], [override]
```

Definition at line 100 of file plugged_layer.h.

8.286.3.11 supportBackwarding()

```
bool nntrainer::internal::PluggedLayer::supportBackwarding ( ) const  [inline], [override]
```

Definition at line 133 of file plugged_layer.h.

8.286.3.12 supportInPlace()

```
bool nntrainer::internal::PluggedLayer::supportInPlace ( ) const  [inline], [override]
```

Definition at line 123 of file plugged_layer.h.

The documentation for this class was generated from the following file:

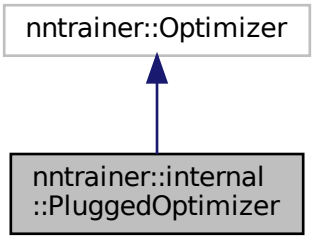
- layers/plugged_layer.h

8.287 nntrainer::internal::PluggedOptimizer Class Reference

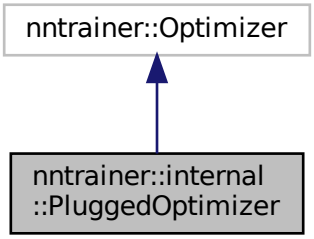
Plugged optimizer class.

`#include <plugged_optimizer.h>`

Inheritance diagram for nntrainer::internal::PluggedOptimizer:



Collaboration diagram for nntrainer::internal::PluggedOptimizer:



Public Member Functions

- PluggedOptimizer (const nntrainer::OptimizerPluggable *pluggable)
  Construct a new Plugged Optimizer object.
- ~PluggedOptimizer () override
  Destroy the Plugged Optimizer object.
- PluggedOptimizer (PluggedOptimizer &&rhs) noexcept=default
  Move Construct Plugged Optimizer object.
- PluggedOptimizer & operator= (PluggedOptimizer &&rhs)=default
  Move assign Plugged Optimizer Object.
- double getDefaultLearningRate () const override
- void applyGradient (RunOptimizerContext &context) override

apply gradient to weight

- void setProperty (const std::vector< std::string > &values) override

  set Optimizer Parameters

- void finalize () override

  finalize optimizer.

- void read (std::ifstream &file) override

  Read Training optimizer parameters from file.

- void save (std::ofstream &file) override

  Save Training optimizer parameters from file.

- virtual std::vector< TensorDim > getOptimizerVariableDim (const TensorDim &dim) override

  Get dimension of extra variables if the optimizer needs any.

- const std::string getType () const override

  get Optimizer Type

8.287.1 Detailed Description

Plugged optimizer class.

Definition at line 29 of file plugged_optimizer.h.

8.287.2 Constructor & Destructor Documentation

8.287.2.1 PluggedOptimizer() [1/2]

```
nntrainer::internal::PluggedOptimizer::PluggedOptimizer (
            const nntrainer::OptimizerPluggable * pluggable )  [inline]
```

Construct a new Plugged Optimizer object.

Parameters

pluggable	OptimizerPluggable structure from the symbol

Definition at line 36 of file plugged_optimizer.h.

8.287.2.2 ~PluggedOptimizer()

```
nntrainer::internal::PluggedOptimizer::~PluggedOptimizer ( )  [inline], [override]
```

Destroy the Plugged Optimizer object.

Definition at line 48 of file plugged_optimizer.h.

8.287.2.3 PluggedOptimizer() [2/2]

```
nntrainer::internal::PluggedOptimizer::PluggedOptimizer (
            PluggedOptimizer && rhs )  [default], [noexcept]
```

Move Construct Plugged Optimizer object.

Parameters

rhs	optimizer to move

8.287.3 Member Function Documentation

8.287.3.1 applyGradient()

```
void nntrainer::internal::PluggedOptimizer::applyGradient (
            RunOptimizerContext & context )  [inline], [override]
```

apply gradient to weight

Parameters

in	*context*	Optimizer context

Definition at line 76 of file plugged_optimizer.h.

8.287.3.2 getDefaultLearningRate()

```
double nntrainer::internal::PluggedOptimizer::getDefaultLearningRate ( ) const  [inline],
[override]
```

Definition at line 69 of file plugged_optimizer.h.

8.287.3.3 getOptimizerVariableDim()

```
virtual std::vector<TensorDim> nntrainer::internal::PluggedOptimizer::getOptimizerVariableDim
(
            const TensorDim & dim )  [inline], [override], [virtual]
```

Get dimension of extra variables if the optimizer needs any.

Parameters

dim	Dimension of tensor to be added as a optimizer variable

Returns

Vector of dimensions

Definition at line 113 of file plugged_optimizer.h.

8.287.3.4 getType()

```
const std::string nntrainer::internal::PluggedOptimizer::getType ( ) const  [inline], [override]
```

get Optimizer Type

Return values

Optimizer	type

Definition at line 121 of file plugged_optimizer.h.

8.287.3.5 operator=()

```
PluggedOptimizer& nntrainer::internal::PluggedOptimizer::operator= (
            PluggedOptimizer && rhs )  [default]
```

Move assign Plugged Optimizer Object.

Parameters

rhs	optimizer to move

Returns

PluggedOptimizer& ∗this

8.287.3.6 read()

```
void nntrainer::internal::PluggedOptimizer::read (
            std::ifstream & file )  [inline], [override]
```

Read Training optimizer parameters from file.

Parameters

in	*file*	input stream file

Definition at line 99 of file plugged_optimizer.h.

8.287.3.7 save()

```
void nntrainer::internal::PluggedOptimizer::save (
            std::ofstream & file )  [inline], [override]
```

Save Training optimizer parameters from file.

Parameters

in	*file*	output stream file

Definition at line 105 of file plugged_optimizer.h.

8.287.3.8 setProperty()

```
void nntrainer::internal::PluggedOptimizer::setProperty (
            const std::vector< std::string > & values )  [inline], [override]
```

set Optimizer Parameters

Parameters

in	*values*	Optimizer Parameter list

Return values

ML_ERROR_NONE	Successful.
ML_ERROR_INVALID_PARAMETER	invalid parameter.

Definition at line 86 of file plugged_optimizer.h.

The documentation for this class was generated from the following file:

- optimizers/plugged_optimizer.h

8.288 tflite::Pool2DOptionsBuilder Struct Reference

Public Types

- typedef Pool2DOptions **Table**

Public Member Functions

- void **add_padding** (tflite::Padding padding)
- void **add_stride_w** (int32_t stride_w)
- void **add_stride_h** (int32_t stride_h)
- void **add_filter_width** (int32_t filter_width)
- void **add_filter_height** (int32_t filter_height)
- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- **Pool2DOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- Pool2DOptionsBuilder & **operator=** (const Pool2DOptionsBuilder &)
- flatbuffers::Offset< Pool2DOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.288.1 Detailed Description

Definition at line 2808 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

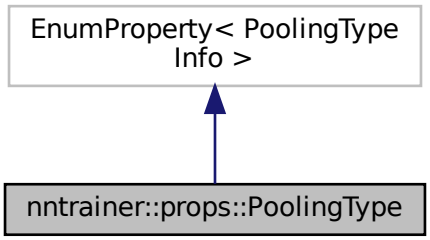
- compiler/tf_schema_generated.h

8.289 nntrainer::props::PoolingType Class Reference

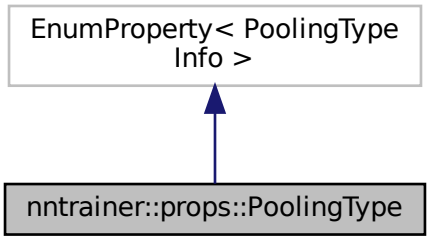
Pooling Type Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::PoolingType:



Collaboration diagram for nntrainer::props::PoolingType:



Public Types

- using **prop_tag** = enum_class_prop_tag

Static Public Attributes

- static constexpr const char * **key** = "pooling"

8.289.1 Detailed Description

Pooling Type Enumeration Information.

Definition at line 1070 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.290 nntrainer::props::PoolingTypeInfo Struct Reference

Enumeration of pooling type.

`#include <common_properties.h>`

Public Types

- enum Enum {
  max = 0, **average** = 1, **global_max** = 2, **global_average** = 3,
  unknown = 4 }

  Pooling operation type class.

Static Public Attributes

- static constexpr std::initializer_list< Enum > **EnumList**
- static constexpr const char * **EnumStr** [ ]

8.290.1 Detailed Description

Enumeration of pooling type.

Definition at line 1047 of file common_properties.h.

8.290.2 Member Data Documentation

8.290.2.1 EnumList

```
constexpr std::initializer_list<Enum> nntrainer::props::PoolingTypeInfo::EnumList  [static],
[constexpr]
```

Initial value:

```
= {
    Enum::max, Enum::average, Enum::global_max, Enum::global_average,
    Enum::unknown}
```

Definition at line 1058 of file common_properties.h.

8.290.2.2 EnumStr

```
constexpr const char* nntrainer::props::PoolingTypeInfo::EnumStr[]  [static], [constexpr]
```

Initial value:

```
= {"max", "average", "global_max",
                                          "global_average", "unknown"}
```

Definition at line 1062 of file common_properties.h.

The documentation for this struct was generated from the following file:

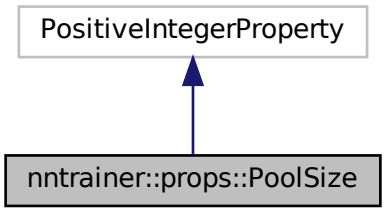
- layers/common_properties.h

8.291 nntrainer::props::PoolSize Class Reference

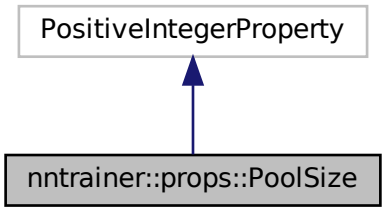
PoolSize property, pool size is used to measure the pooling size.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::PoolSize:



Collaboration diagram for nntrainer::props::PoolSize:



Public Types

- using prop_tag = uint_prop_tag

Public Member Functions

- PoolSize ()

  Construct a new PoolSize object.
- PoolSize (unsigned int value)

  Construct a new PoolSize object with default value.

Static Public Attributes

- static constexpr const char * key = "pool_size"

8.291.1 Detailed Description

PoolSize property, pool size is used to measure the pooling size.

Definition at line 341 of file common_properties.h.

8.291.2 Member Typedef Documentation

8.291.2.1 prop_tag

```
using nntrainer::props::PoolSize::prop_tag = uint_prop_tag
```

property type

Definition at line 355 of file common_properties.h.

8.291.3 Constructor & Destructor Documentation

8.291.3.1 PoolSize() [1/2]

```
nntrainer::props::PoolSize::PoolSize ( )  [inline]
```

Construct a new PoolSize object.

Definition at line 347 of file common_properties.h.

8.291.3.2 PoolSize() [2/2]

```
nntrainer::props::PoolSize::PoolSize (
            unsigned int value )
```

Construct a new PoolSize object with default value.

Definition at line 107 of file common_properties.cpp.

8.291.4 Member Data Documentation

8.291.4.1 key

```
constexpr const char* nntrainer::props::PoolSize::key = "pool_size"  [static], [constexpr]
```

unique key to access

Definition at line 354 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.292 tflite::PowOptionsBuilder Struct Reference

Public Types

- typedef PowOptions **Table**

Public Member Functions

- **PowOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- PowOptionsBuilder & **operator=** (const PowOptionsBuilder &)
- flatbuffers::Offset< PowOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.292.1 Detailed Description

Definition at line 5722 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

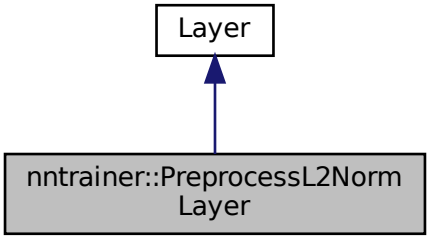
- compiler/tf_schema_generated.h

8.293 nntrainer::PreprocessL2NormLayer Class Reference

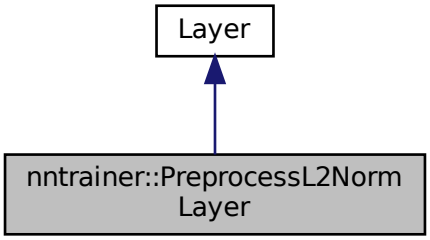
Layer class that l2normalizes a feature vector.

`#include <preprocess_l2norm_layer.h>`

Inheritance diagram for nntrainer::PreprocessL2NormLayer:



Collaboration diagram for nntrainer::PreprocessL2NormLayer:



Public Member Functions

- PreprocessL2NormLayer ()
  Construct a new L2norm Layer object that normlizes given feature with l2norm.
- PreprocessL2NormLayer (PreprocessL2NormLayer &&rhs) noexcept=default
  Move constructor.
- PreprocessL2NormLayer & operator= (PreprocessL2NormLayer &&rhs)=default
  Move assignment operator. @parma[in] rhs PreprocessL2NormLayer to be moved.
- ~PreprocessL2NormLayer ()
  Destroy the Centering Layer object.
- void finalize (InitLayerContext &context) override
- void forwarding (RunLayerContext &context, bool training) override

- void calcDerivative (RunLayerContext &context) override
- bool supportBackwarding () const override

  supportBackwarding() const
- void exportTo (Exporter &exporter, const ml::train::ExportMethods &method) const override
- const std::string getType () const override
- void setProperty (const std::vector< std::string > &values) override

Static Public Attributes

- static const std::string **type** = "preprocess_l2norm"

8.293.1 Detailed Description

Layer class that l2normalizes a feature vector.

Definition at line 28 of file preprocess_l2norm_layer.h.

8.293.2 Constructor & Destructor Documentation

8.293.2.1 PreprocessL2NormLayer()

```
nntrainer::PreprocessL2NormLayer::PreprocessL2NormLayer (
            PreprocessL2NormLayer && rhs )  [default], [noexcept]
```

Move constructor.

Parameters

in	*PreprocessL2NormLayer*	&&

8.293.2.2 ~PreprocessL2NormLayer()

```
nntrainer::PreprocessL2NormLayer::~PreprocessL2NormLayer ( )  [inline]
```

Destroy the Centering Layer object.

Definition at line 52 of file preprocess_l2norm_layer.h.

8.293.3 Member Function Documentation

8.293.3.1 calcDerivative()

```
void nntrainer::PreprocessL2NormLayer::calcDerivative (
            RunLayerContext & context )  [override]
```

Definition at line 53 of file preprocess_l2norm_layer.cpp.

8.293.3.2 exportTo()

```
void nntrainer::PreprocessL2NormLayer::exportTo (
            Exporter & exporter,
            const ml::train::ExportMethods & method ) const  [inline], [override]
```

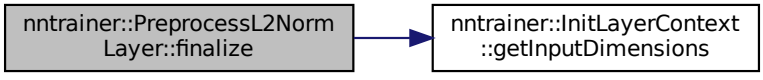
Definition at line 78 of file preprocess_l2norm_layer.h.

8.293.3.3 finalize()

```
void nntrainer::PreprocessL2NormLayer::finalize (
            InitLayerContext & context )  [override]
```

Definition at line 28 of file preprocess_l2norm_layer.cpp.

Here is the call graph for this function:

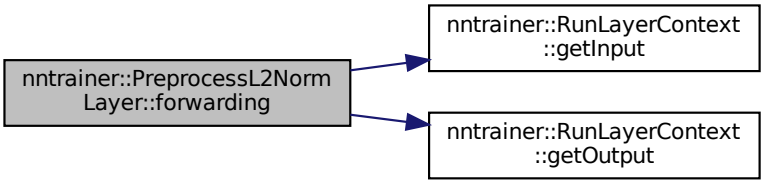


8.293.3.4 forwarding()

```
void nntrainer::PreprocessL2NormLayer::forwarding (
            RunLayerContext & context,
            bool training )  [override]
```

Definition at line 40 of file preprocess_l2norm_layer.cpp.

Here is the call graph for this function:



8.293.3.5 getType()

```
const std::string nntrainer::PreprocessL2NormLayer::getType ( ) const  [inline], [override]
```

Definition at line 84 of file preprocess_l2norm_layer.h.

8.293.3.6 setProperty()

```
void nntrainer::PreprocessL2NormLayer::setProperty (
            const std::vector< std::string > & values )  [override]
```

Definition at line 58 of file preprocess_l2norm_layer.cpp.

8.293.3.7 supportBackwarding()

```
bool nntrainer::PreprocessL2NormLayer::supportBackwarding ( ) const  [inline], [override]
```

supportBackwarding() const

supportBackwarding() const

Definition at line 72 of file preprocess_l2norm_layer.h.

The documentation for this class was generated from the following files:

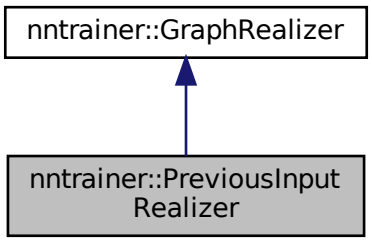
- layers/preprocess_l2norm_layer.h
- layers/preprocess_l2norm_layer.cpp

8.294 nntrainer::PreviousInputRealizer Class Reference

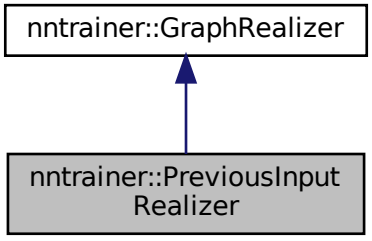
Add default inputs if input connection is empty. if a node is identified as input with *identified_input* by user or a node has input_shape property, adding input behavior is skipped.

`#include <previous_input_realizer.h>`

Inheritance diagram for nntrainer::PreviousInputRealizer:



Collaboration diagram for nntrainer::PreviousInputRealizer:



Public Member Functions

- PreviousInputRealizer (const std::vector< Connection > &identified_inputs)

  Construct a new Previous Input Realizer object.
- ~PreviousInputRealizer ()

  Destroy the Graph Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  graph realizer creates a new graph based on the reference

8.294.1 Detailed Description

Add default inputs if input connection is empty. if a node is identified as input with *identified_input* by user or a node has input_shape property, adding input behavior is skipped.

Definition at line 29 of file previous_input_realizer.h.

8.294.2 Constructor & Destructor Documentation

8.294.2.1 PreviousInputRealizer()

```
nntrainer::PreviousInputRealizer::PreviousInputRealizer (
            const std::vector< Connection > & identified_inputs )
```

Construct a new Previous Input Realizer object.

Parameters

identified_inputs	node that is identified as an input, this must not connect to other nodes automatically

Definition at line 25 of file previous_input_realizer.cpp.

8.294.2.2 ~PreviousInputRealizer()

```
nntrainer::PreviousInputRealizer::~PreviousInputRealizer ( )
```

Destroy the Graph Realizer object.

Definition at line 29 of file previous_input_realizer.cpp.

8.294.3 Member Function Documentation

8.294.3.1 realize()

```
GraphRepresentation nntrainer::PreviousInputRealizer::realize (
            const GraphRepresentation & reference ) [override], [virtual]
```

graph realizer creates a new graph based on the reference

for node has input connection, below function determines if the node should be input node or add input_layers from previous layer

Implements nntrainer::GraphRealizer.

Definition at line 32 of file previous_input_realizer.cpp.

The documentation for this class was generated from the following files:

- compiler/previous_input_realizer.h
- compiler/previous_input_realizer.cpp

8.295 nntrainer::profile::ProfileEventData Struct Reference

Data for each profile event.

`#include <profiler.h>`

Public Member Functions

- ProfileEventData (int item, size_t cur, size_t total, std::string str, std::chrono::microseconds dur)

  Construct a new ProfileEventData struct.
- ProfileEventData (int item, size_t cur, size_t total, std::string str, std::chrono::microseconds dur, std::string policy, bool swap)

  Construct a new ProfileEventData struct.

Public Attributes

- int **time_item**
- size_t **alloc_current**
- size_t **alloc_total**
- std::string **cache_policy**
- bool **cache_swap**
- std::string **event_str**
- std::chrono::microseconds **duration**

8.295.1 Detailed Description

Data for each profile event.

Definition at line 99 of file profiler.h.

8.295.2 Constructor & Destructor Documentation

8.295.2.1 ProfileEventData() [1/2]

```
nntrainer::profile::ProfileEventData::ProfileEventData (
            int item,
            size_t cur,
            size_t total,
            std::string str,
            std::chrono::microseconds dur )  [inline]
```

Construct a new ProfileEventData struct.

Definition at line 105 of file profiler.h.

8.295.2.2 ProfileEventData() [2/2]

```
nntrainer::profile::ProfileEventData::ProfileEventData (
            int item,
            size_t cur,
            size_t total,
            std::string str,
            std::chrono::microseconds dur,
            std::string policy,
            bool swap )  [inline]
```

Construct a new ProfileEventData struct.

Definition at line 118 of file profiler.h.

The documentation for this struct was generated from the following file:

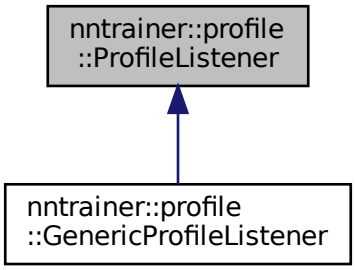
- utils/profiler.h

8.296 nntrainer::profile::ProfileListener Class Reference

Generic profile listener class to attach to a profiler, this can be inherited to create a custom profile listener.

`#include <profiler.h>`

Inheritance diagram for nntrainer::profile::ProfileListener:



Public Member Functions

- ProfileListener ()=default
  Construct a new Profile Listener object.
- virtual ~ProfileListener () noexcept=default
  Destroy the Base Profile Listener object.
- virtual void notify (PROFILE_EVENT event, const std::shared_ptr< ProfileEventData > data)=0
  A callback function to be called from a profiler.
- virtual void reset (const int time_item, const std::string &str)=0
  resets the listener to the inital state for a particular key
- virtual const std::chrono::microseconds result (const int time_item)=0
  get the latest result of a event
- virtual void report (std::ostream &out) const =0
  report the result

8.296.1 Detailed Description

Generic profile listener class to attach to a profiler, this can be inherited to create a custom profile listener.

Definition at line 152 of file profiler.h.

8.296.2 Constructor & Destructor Documentation

8.296.2.1 ProfileListener()

```
nntrainer::profile::ProfileListener::ProfileListener ( )  [explicit], [default]
```

Construct a new Profile Listener object.

8.296.2.2 ~ProfileListener()

```
virtual nntrainer::profile::ProfileListener::~ProfileListener ( )  [virtual], [default], [noexcept]
```

Destroy the Base Profile Listener object.

8.296.3 Member Function Documentation

8.296.3.1 notify()

```
virtual void nntrainer::profile::ProfileListener::notify (
            PROFILE_EVENT event,
            const std::shared_ptr< ProfileEventData > data )  [pure virtual]
```

A callback function to be called from a profiler.

Parameters

event	event type
data	event data

Implemented in nntrainer::profile::GenericProfileListener.

8.296.3.2 report()

```
virtual void nntrainer::profile::ProfileListener::report (
            std::ostream & out ) const  [pure virtual]
```

report the result

Parameters

out	outstream object to make a report

Implemented in nntrainer::profile::GenericProfileListener.

8.296.3.3 reset()

```
virtual void nntrainer::profile::ProfileListener::reset (
            const int time_item,
            const std::string & str )  [pure virtual]
```

resets the listener to the inital state for a particular key

Parameters

time_item	time item which will be reset

Implemented in nntrainer::profile::GenericProfileListener.

8.296.3.4 result()

```
virtual const std::chrono::microseconds nntrainer::profile::ProfileListener::result (
            const int time_item )  [pure virtual]
```

get the latest result of a event

Parameters

time_item	time item to query the result

Returns

const std::chrono::microseconds

Implemented in nntrainer::profile::GenericProfileListener.

The documentation for this class was generated from the following file:

- utils/profiler.h

8.297 nntrainer::profile::Profiler Class Reference

Profiler object.

`#include <profiler.h>`

Public Member Functions

- Profiler ()
  Construct a new Profiler object.
- Profiler (const Profiler &)=delete
  Deleted constructor.
- Profiler & **operator=** (const Profiler &)=delete
- void start (const int time_item)
  start time profile
- void end (const int time_item)
  end time profile and notify to the listeners
- void alloc (const void *ptr, size_t size, const std::string &str, const std::string &policy="", bool swap=false)
  trace memory allocation
- void dealloc (const void *ptr, const std::string &policy="", bool swap=false)
  trace memory de-allocation
- void annotate (const std::string &str)
  add annotation on memory profile data
- void subscribe (std::shared_ptr< ProfileListener > listener, const std::set< int > &time_item={})
  subscribe a listener to the profiler
- void unsubscribe (std::shared_ptr< ProfileListener > listener)
  unsubscribe a listener from the profiler
- int registerTimeItem (const std::string &name)
  registerEvent to record.

Static Public Member Functions

- static Profiler & Global ()
  Get Global Profiler.

8.297.1 Detailed Description

Profiler object.

Definition at line 309 of file profiler.h.

8.297.2 Constructor & Destructor Documentation

8.297.2.1 Profiler() [1/2]

```
nntrainer::profile::Profiler::Profiler ( )  [inline]
```

Construct a new Profiler object.

Definition at line 315 of file profiler.h.

8.297.2.2 Profiler() [2/2]

```
nntrainer::profile::Profiler::Profiler (
            const Profiler &  )  [delete]
```

Deleted constructor.

8.297.3 Member Function Documentation

8.297.3.1 alloc()

```
void nntrainer::profile::Profiler::alloc (
            const void * ptr,
            size_t size,
            const std::string & str,
            const std::string & policy = "",
            bool swap = false )
```

trace memory allocation

Parameters

ptr	allocated memory pointer
size	amount of allocated memory
str	information string

Definition at line 355 of file profiler.cpp.

8.297.3.2 annotate()

```
void nntrainer::profile::Profiler::annotate (
            const std::string & str )
```

add annotation on memory profile data

Parameters

str	annotate message

Definition at line 400 of file profiler.cpp.

8.297.3.3 dealloc()

```
void nntrainer::profile::Profiler::dealloc (
            const void * ptr,
            const std::string & policy = "",
            bool swap = false )
```

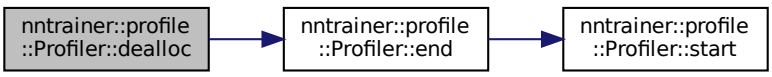
trace memory de-allocation

Parameters

ptr	de-allocated memory pointer

Definition at line 376 of file profiler.cpp.

Here is the call graph for this function:



8.297.3.4 end()

```
void nntrainer::profile::Profiler::end (
            const int time_item )
```

end time profile and notify to the listeners

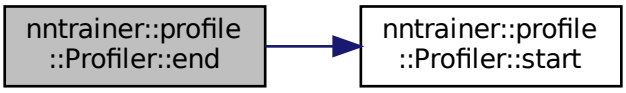
Parameters

time_item	time item to be finished

Todo : consider race condition

Definition at line 280 of file profiler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.297.3.5 Global()

```
Profiler & nntrainer::profile::Profiler::Global ( )  [static]
```

Get Global Profiler.

Returns

Profiler&

Definition at line 257 of file profiler.cpp.

8.297.3.6 registerTimeItem()

```
int nntrainer::profile::Profiler::registerTimeItem (
            const std::string & name )
```

registerEvent to record.

Note

Call to the function shouldn't be inside a critical path

Returns

int return NEGATIVE integer to distinguish from reserved events

Definition at line 341 of file profiler.cpp.

8.297.3.7 start()

```
void nntrainer::profile::Profiler::start (
            const int time_item )
```

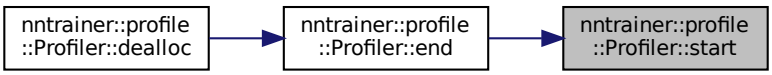
start time profile

Parameters

time_item	time item to be recorded

Definition at line 262 of file profiler.cpp.

Here is the caller graph for this function:



8.297.3.8 subscribe()

```
void nntrainer::profile::Profiler::subscribe (
            std::shared_ptr< ProfileListener > listener,
            const std::set< int > & time_item = {} )
```

subscribe a listener to the profiler

Parameters

listener	listener to register, listener must call unsubscribe on destruction
events	event listeners are subscribing, if empty listener subscribes to all events

Exceptions

std::invalid_argument	if listener is already registered

Definition at line 315 of file profiler.cpp.

8.297.3.9 unsubscribe()

```
void nntrainer::profile::Profiler::unsubscribe (
            std::shared_ptr< ProfileListener > listener )
```

unsubscribe a listener from the profiler

Parameters

listener	listener to unsubscribe

Definition at line 330 of file profiler.cpp.

The documentation for this class was generated from the following files:

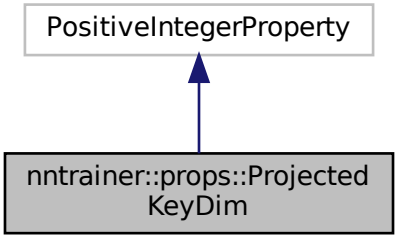
- utils/profiler.h
- utils/profiler.cpp

8.298 nntrainer::props::ProjectedKeyDim Class Reference

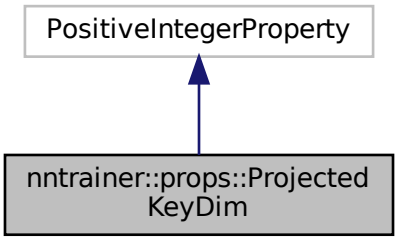
ProjectedKeyDim property, projected key dim per head in multi head attention.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::ProjectedKeyDim:



Collaboration diagram for nntrainer::props::ProjectedKeyDim:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key

8.298.1 Detailed Description

ProjectedKeyDim property, projected key dim per head in multi head attention.

Correspond with key_dim of tensorflow

Definition at line 1190 of file common_properties.h.

8.298.2 Member Typedef Documentation

8.298.2.1 prop_tag

```
using nntrainer::props::ProjectedKeyDim::prop_tag = uint_prop_tag
```

property type

Definition at line 1194 of file common_properties.h.

8.298.3 Member Data Documentation

8.298.3.1 key

```
constexpr const char* nntrainer::props::ProjectedKeyDim::key  [static], [constexpr]
```

Initial value:

```
=
    "projected_key_dim"
```

unique key to access

Definition at line 1192 of file common_properties.h.

The documentation for this class was generated from the following file:

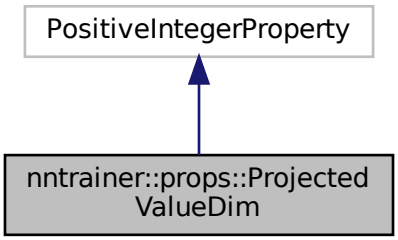
- layers/common_properties.h

8.299 nntrainer::props::ProjectedValueDim Class Reference

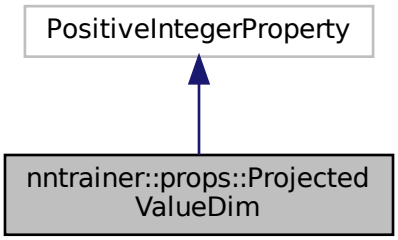
ProjectedValueDim property, projected value dim per head in multi head attention.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::ProjectedValueDim:



Collaboration diagram for nntrainer::props::ProjectedValueDim:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key

8.299.1 Detailed Description

ProjectedValueDim property, projected value dim per head in multi head attention.

Correspond with value_dim of tensorflow

Definition at line 1203 of file common_properties.h.

8.299.2 Member Typedef Documentation

8.299.2.1 prop_tag

```
using nntrainer::props::ProjectedValueDim::prop_tag = uint_prop_tag
```

property type

Definition at line 1207 of file common_properties.h.

8.299.3 Member Data Documentation

8.299.3.1 key

```
constexpr const char* nntrainer::props::ProjectedValueDim::key  [static], [constexpr]
```

Initial value:

```
=
    "projected_value_dim"
```

unique key to access

Definition at line 1205 of file common_properties.h.

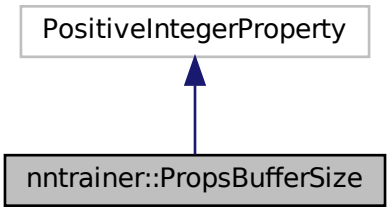
The documentation for this class was generated from the following file:

- layers/common_properties.h

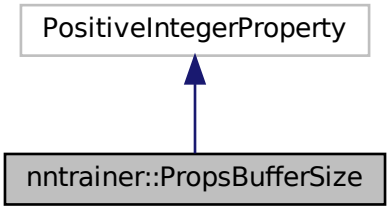
8.300 nntrainer::PropsBufferSize Class Reference

Props containing buffer size value.

Inheritance diagram for nntrainer::PropsBufferSize:



Collaboration diagram for nntrainer::PropsBufferSize:



Public Types

- using prop_tag = uint_prop_tag

Public Member Functions

- PropsBufferSize (unsigned int value=1)

  Construct a new props min object with a default value.

Static Public Attributes

- static constexpr const char * key = "buffer_size"

8.300.1 Detailed Description

Props containing buffer size value.

Definition at line 49 of file databuffer.cpp.

8.300.2 Member Typedef Documentation

8.300.2.1 prop_tag

```
using nntrainer::PropsBufferSize::prop_tag = uint_prop_tag
```

property type

Definition at line 58 of file databuffer.cpp.

8.300.3 Constructor & Destructor Documentation

8.300.3.1 PropsBufferSize()

```
nntrainer::PropsBufferSize::PropsBufferSize (
            unsigned int value = 1 )  [inline]
```

Construct a new props min object with a default value.

Parameters

value	default value

Definition at line 56 of file databuffer.cpp.

8.300.4 Member Data Documentation

8.300.4.1 key

```
constexpr const char* nntrainer::PropsBufferSize::key = "buffer_size"  [static], [constexpr]
```

unique key to access

Definition at line 57 of file databuffer.cpp.

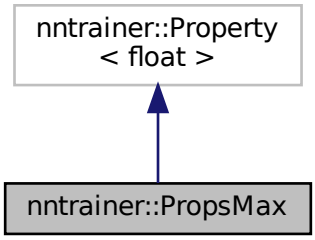
The documentation for this class was generated from the following file:

- dataset/databuffer.cpp

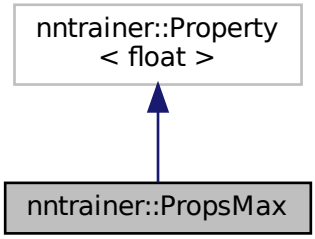
8.301 nntrainer::PropsMax Class Reference

Props containing max value.

Inheritance diagram for nntrainer::PropsMax:



Collaboration diagram for nntrainer::PropsMax:



Public Types

- using prop_tag = float_prop_tag

Public Member Functions

- PropsMax (float value=1.0f)

  Construct a new props max object with a default value.

Static Public Attributes

- static constexpr const char * key = "max"

8.301.1 Detailed Description

Props containing max value.

Definition at line 41 of file random_data_producers.cpp.

8.301.2 Member Typedef Documentation

8.301.2.1 prop_tag

```
using nntrainer::PropsMax::prop_tag = float_prop_tag
```

property type

Definition at line 50 of file random_data_producers.cpp.

8.301.3 Constructor & Destructor Documentation

8.301.3.1 PropsMax()

```
nntrainer::PropsMax::PropsMax (
            float value = 1.0f )  [inline]
```

Construct a new props max object with a default value.

Parameters

value	default value

Definition at line 48 of file random_data_producers.cpp.

8.301.4 Member Data Documentation

8.301.4.1 key

```
constexpr const char* nntrainer::PropsMax::key = "max"  [static], [constexpr]
```

unique key to access

Definition at line 49 of file random_data_producers.cpp.

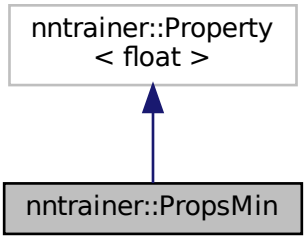
The documentation for this class was generated from the following file:

- dataset/random_data_producers.cpp

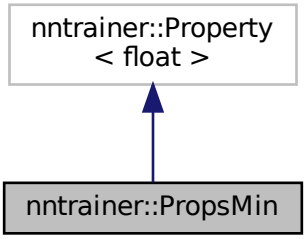
8.302 nntrainer::PropsMin Class Reference

Props containing min value.

Inheritance diagram for nntrainer::PropsMin:



Collaboration diagram for nntrainer::PropsMin:



Public Types

- using prop_tag = float_prop_tag

Public Member Functions

- PropsMin (float value=0.0f)

  Construct a new props min object with a default value.

Static Public Attributes

- static constexpr const char * key = "min"

8.302.1 Detailed Description

Props containing min value.

Definition at line 25 of file random_data_producers.cpp.

8.302.2 Member Typedef Documentation

8.302.2.1 prop_tag

```
using nntrainer::PropsMin::prop_tag = float_prop_tag
```

property type

Definition at line 34 of file random_data_producers.cpp.

8.302.3 Constructor & Destructor Documentation

8.302.3.1 PropsMin()

```
nntrainer::PropsMin::PropsMin (
            float value = 0.0f )  [inline]
```

Construct a new props min object with a default value.

Parameters

value	default value

Definition at line 32 of file random_data_producers.cpp.

8.302.4 Member Data Documentation

8.302.4.1 key

```
constexpr const char* nntrainer::PropsMin::key = "min"  [static], [constexpr]
```

unique key to access

Definition at line 33 of file random_data_producers.cpp.

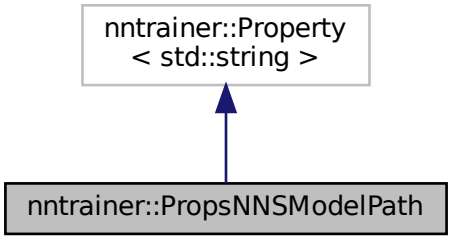
The documentation for this class was generated from the following file:

- dataset/random_data_producers.cpp

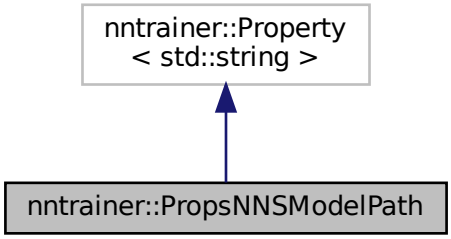
8.303 nntrainer::PropsNNSModelPath Class Reference

NNSModelPath property.

Inheritance diagram for nntrainer::PropsNNSModelPath:



Collaboration diagram for nntrainer::PropsNNSModelPath:



Public Types

- using prop_tag = str_prop_tag

Public Member Functions

- bool isValid (const std::string &v) const override

  check is valid

Static Public Attributes

- static constexpr const char * key = "model_path"

8.303.1 Detailed Description

NNSModelPath property.

Definition at line 31 of file nnstreamer_layer.cpp.

8.303.2 Member Typedef Documentation

8.303.2.1 prop_tag

```
using nntrainer::PropsNNSModelPath::prop_tag = str_prop_tag
```

property type

Definition at line 34 of file nnstreamer_layer.cpp.

8.303.3 Member Function Documentation

8.303.3.1 isValid()

```
bool nntrainer::PropsNNSModelPath::isValid (
            const std::string & v ) const  [override]
```

check is valid

Parameters

v	value to check

Returns

bool true if valid

Note

this might not be good validation strategy, as it can support directory or even not initiated path??

Definition at line 44 of file nnstreamer_layer.cpp.

8.303.4 Member Data Documentation

8.303.4.1 key

```
constexpr const char* nntrainer::PropsNNSModelPath::key = "model_path"  [static], [constexpr]
```

unique key to access

Definition at line 33 of file nnstreamer_layer.cpp.

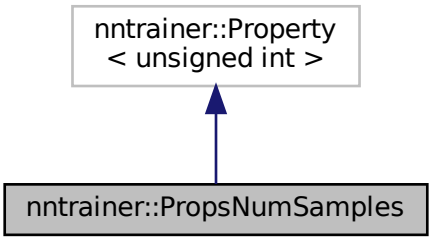
The documentation for this class was generated from the following file:

- layers/nnstreamer_layer.cpp

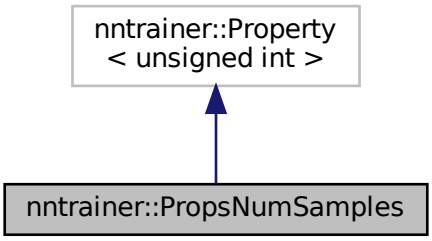
8.304 nntrainer::PropsNumSamples Class Reference

Props containing number of samples A random data producer has theoretical size. number of samples is used to set theoretical size of the random data producer's data size.

Inheritance diagram for nntrainer::PropsNumSamples:



Collaboration diagram for nntrainer::PropsNumSamples:



Public Types

- using prop_tag = uint_prop_tag

Public Member Functions

- PropsNumSamples (unsigned int value=512)

  Construct a new props data size object with a default value.

Static Public Attributes

- static constexpr const char * key = "num_samples"

8.304.1 Detailed Description

Props containing number of samples A random data producer has theoretical size. number of samples is used to set theoretical size of the random data producer's data size.

Definition at line 59 of file random_data_producers.cpp.

8.304.2 Member Typedef Documentation

8.304.2.1 prop_tag

```
using nntrainer::PropsNumSamples::prop_tag = uint_prop_tag
```

property type

Definition at line 69 of file random_data_producers.cpp.

8.304.3 Constructor & Destructor Documentation

8.304.3.1 PropsNumSamples()

```
nntrainer::PropsNumSamples::PropsNumSamples (
            unsigned int value = 512 )  [inline]
```

Construct a new props data size object with a default value.

Parameters

value	default value

Definition at line 66 of file random_data_producers.cpp.

8.304.4 Member Data Documentation

8.304.4.1 key

```
constexpr const char* nntrainer::PropsNumSamples::key = "num_samples"  [static], [constexpr]
```

unique key to access

Definition at line 68 of file random_data_producers.cpp.

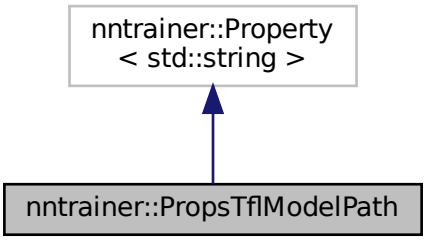
The documentation for this class was generated from the following file:

- dataset/random_data_producers.cpp

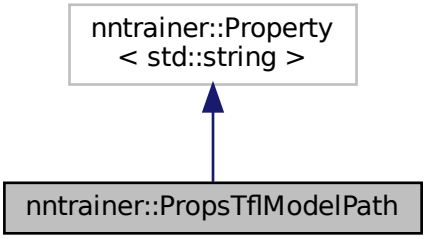
8.305 nntrainer::PropsTflModelPath Class Reference

TflModelPath property.

Inheritance diagram for nntrainer::PropsTflModelPath:



Collaboration diagram for nntrainer::PropsTflModelPath:



Public Types

- using prop_tag = str_prop_tag

Public Member Functions

- bool isValid (const std::string &v) const override

  check is valid

Static Public Attributes

- static constexpr const char * key = "model_path"
- static constexpr const char **ending** [ ] = ".tflite"
- static constexpr unsigned int **ending_len** = 7

8.305.1 Detailed Description

TflModelPath property.

Definition at line 27 of file tflite_layer.cpp.

8.305.2 Member Typedef Documentation

8.305.2.1 prop_tag

```
using nntrainer::PropsTflModelPath::prop_tag = str_prop_tag
```

property type

Definition at line 30 of file tflite_layer.cpp.

8.305.3 Member Function Documentation

8.305.3.1 isValid()

```
bool nntrainer::PropsTflModelPath::isValid (
            const std::string & v ) const  [override]
```

check is valid

Parameters

v	value to check

Returns

bool true if valid

check if path ends with .tflite

Definition at line 43 of file tflite_layer.cpp.

8.305.4 Member Data Documentation

8.305.4.1 key

```
constexpr const char* nntrainer::PropsTflModelPath::key = "model_path"  [static], [constexpr]
```

unique key to access

Definition at line 29 of file tflite_layer.cpp.

The documentation for this class was generated from the following file:

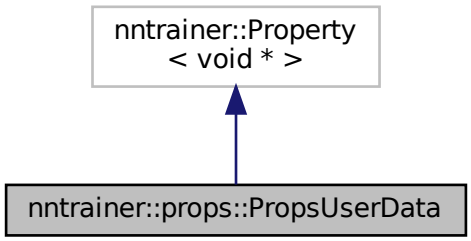
- layers/tflite_layer.cpp

8.306 nntrainer::props::PropsUserData Class Reference

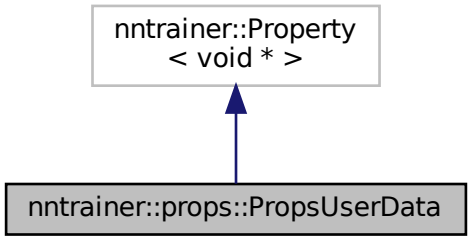
User data props.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::PropsUserData:



Collaboration diagram for nntrainer::props::PropsUserData:



Public Types

- using **prop_tag** = ptr_prop_tag

Public Member Functions

- **PropsUserData** (void ∗user_data)

Static Public Attributes

- static constexpr const char ∗ **key** = "user_data"

8.306.1 Detailed Description

User data props.

Definition at line 1325 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.307 tflite::QuantizationDetailsTraits< T > Struct Template Reference

Static Public Attributes

- static const QuantizationDetails **enum_value** = QuantizationDetails_NONE

8.307.1 Detailed Description

template<typename T>
struct tflite::QuantizationDetailsTraits< T >

Definition at line 455 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.308 tflite::QuantizationDetailsTraits< tflite::CustomQuantization > Struct Reference

Static Public Attributes

- static const QuantizationDetails **enum_value** = QuantizationDetails_CustomQuantization

8.308.1 Detailed Description

Definition at line 459 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.309 tflite::QuantizationParametersBuilder Struct Reference

Public Types

- typedef QuantizationParameters **Table**

Public Member Functions

- void **add_min** (flatbuffers::Offset< flatbuffers::Vector< float >> min)
- void **add_max** (flatbuffers::Offset< flatbuffers::Vector< float >> max)
- void **add_scale** (flatbuffers::Offset< flatbuffers::Vector< float >> scale)
- void **add_zero_point** (flatbuffers::Offset< flatbuffers::Vector< int64_t >> zero_point)
- void **add_details_type** (tflite::QuantizationDetails details_type)
- void **add_details** (flatbuffers::Offset< void > details)
- void **add_quantized_dimension** (int32_t quantized_dimension)
- **QuantizationParametersBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- QuantizationParametersBuilder & **operator=** (const QuantizationParametersBuilder &)
- flatbuffers::Offset< QuantizationParameters > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.309.1 Detailed Description

Definition at line 2076 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.310 tflite::QuantizeOptionsBuilder Struct Reference

Public Types

- typedef QuantizeOptions **Table**

Public Member Functions

- **QuantizeOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- QuantizeOptionsBuilder & **operator=** (const QuantizeOptionsBuilder &)
- flatbuffers::Offset< QuantizeOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.310.1 Detailed Description

Definition at line 6658 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

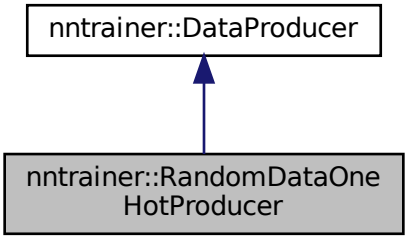
- compiler/tf_schema_generated.h

8.311 nntrainer::RandomDataOneHotProducer Class Reference

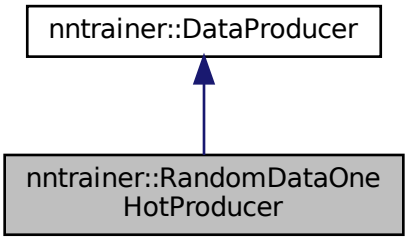
RandomDataProducer which generates a onehot vector as a label.

`#include <random_data_producers.h>`

Inheritance diagram for nntrainer::RandomDataOneHotProducer:



Collaboration diagram for nntrainer::RandomDataOneHotProducer:



Public Member Functions

- RandomDataOneHotProducer ()

  Construct a new Random Data One Hot Producer object.
- ∼RandomDataOneHotProducer ()

  Destroy the Random Data One Hot Producer object.
- const std::string getType () const override

  Get the producer type.
- bool isMultiThreadSafe () const override

  denote if given producer is thread safe and can be parallelized.
- void setProperty (const std::vector< std::string > &properties) override
- DataProducer::Generator finalize (const std::vector< TensorDim > &input_dims, const std::vector< Tensor↩ Dim > &label_dims, void ∗user_data=nullptr) override
- unsigned int size (const std::vector< TensorDim > &input_dims, const std::vector< TensorDim > &label_↩ dims) const override

Static Public Attributes

- static const std::string **type** = "random_data_one_hot"

Additional Inherited Members

8.311.1 Detailed Description

RandomDataProducer which generates a onehot vector as a label.

Definition at line 33 of file random_data_producers.h.

8.311.2 Constructor & Destructor Documentation

8.311.2.1 RandomDataOneHotProducer()

```
nntrainer::RandomDataOneHotProducer::RandomDataOneHotProducer ( )
```

Construct a new Random Data One Hot Producer object.

Definition at line 72 of file random_data_producers.cpp.

8.311.2.2 ∼RandomDataOneHotProducer()

```
nntrainer::RandomDataOneHotProducer::∼RandomDataOneHotProducer ( )
```

Destroy the Random Data One Hot Producer object.

Definition at line 75 of file random_data_producers.cpp.

8.311.3 Member Function Documentation

8.311.3.1 finalize()

```
DataProducer::Generator nntrainer::RandomDataOneHotProducer::finalize (
            const std::vector< TensorDim > & input_dims,
            const std::vector< TensorDim > & label_dims,
            void * user_data = nullptr )  [override], [virtual]
```

check if the given producer is ready to finalize

Todo expand this to non onehot case

Todo move this to higher order component

prepare states for the generator

DataProducer::Generator

Reimplemented from nntrainer::DataProducer.

Definition at line 94 of file random_data_producers.cpp.

8.311.3.2 getType()

```
const std::string nntrainer::RandomDataOneHotProducer::getType ( ) const  [override], [virtual]
```

Get the producer type.

Returns

const std::string type representation

Implements nntrainer::DataProducer.

Definition at line 77 of file random_data_producers.cpp.

8.311.3.3 isMultiThreadSafe()

```
bool nntrainer::RandomDataOneHotProducer::isMultiThreadSafe ( ) const  [override], [virtual]
```

denote if given producer is thread safe and can be parallelized.

Note

if size() == SIZE_UNDEFIEND, thread safe shall be false

Returns

bool true if thread safe.

Todo make this true, it is needed to test multiple worker scenario

Reimplemented from nntrainer::DataProducer.

Definition at line 81 of file random_data_producers.cpp.

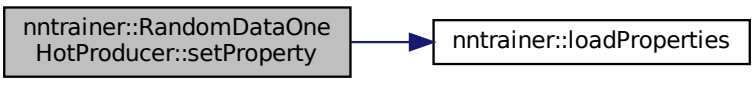
8.311.3.4 setProperty()

```
void nntrainer::RandomDataOneHotProducer::setProperty (
            const std::vector< std::string > & properties )  [override], [virtual]
```

Reimplemented from nntrainer::DataProducer.

Definition at line 86 of file random_data_producers.cpp.

Here is the call graph for this function:



8.311.3.5 size()

```
unsigned int nntrainer::RandomDataOneHotProducer::size (
            const std::vector< TensorDim > & input_dims,
            const std::vector< TensorDim > & label_dims ) const  [override], [virtual]
```

Reimplemented from nntrainer::DataProducer.

Definition at line 155 of file random_data_producers.cpp.

The documentation for this class was generated from the following files:

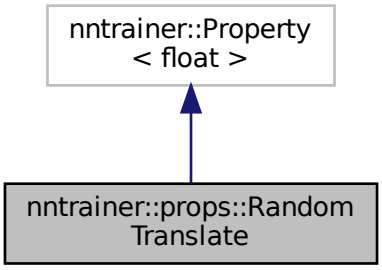
- dataset/random_data_producers.h
- dataset/random_data_producers.cpp

8.312 nntrainer::props::RandomTranslate Class Reference

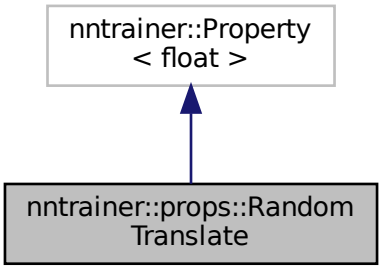
TranslationFactor property, this defines how far the image is translated.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::RandomTranslate:



Collaboration diagram for nntrainer::props::RandomTranslate:



Public Types

- using prop_tag = float_prop_tag

Public Member Functions

- void set (const float &value) override
  setter

Static Public Attributes

- static constexpr const char * key

8.312.1 Detailed Description

TranslationFactor property, this defines how far the image is translated.

Definition at line 531 of file common_properties.h.

8.312.2 Member Typedef Documentation

8.312.2.1 prop_tag

```
using nntrainer::props::RandomTranslate::prop_tag = float_prop_tag
```

property type

Definition at line 536 of file common_properties.h.

8.312.3 Member Function Documentation

8.312.3.1 set()

```
void nntrainer::props::RandomTranslate::set (
            const float & value )  [override]
```

setter

Parameters

value	value to set

Definition at line 55 of file common_properties.cpp.

8.312.4 Member Data Documentation

8.312.4.1 key

```
constexpr const char* nntrainer::props::RandomTranslate::key  [static], [constexpr]
```

Initial value:

```
=
    "random_translate"
```

unique key to access

Definition at line 534 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.313 tflite::RangeOptionsBuilder Struct Reference

Public Types

- typedef RangeOptions **Table**

Public Member Functions

- **RangeOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- RangeOptionsBuilder & **operator=** (const RangeOptionsBuilder &)
- flatbuffers::Offset< RangeOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.313.1 Detailed Description

Definition at line 6270 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.314 tflite::RankOptionsBuilder Struct Reference

Public Types

- typedef RankOptions **Table**

Public Member Functions

- **RankOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- RankOptionsBuilder & **operator=** (const RankOptionsBuilder &)
- flatbuffers::Offset< RankOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.314.1 Detailed Description

Definition at line 5692 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

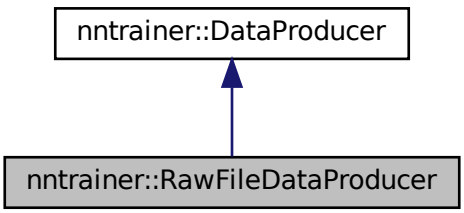
- compiler/tf_schema_generated.h

8.315 nntrainer::RawFileDataProducer Class Reference

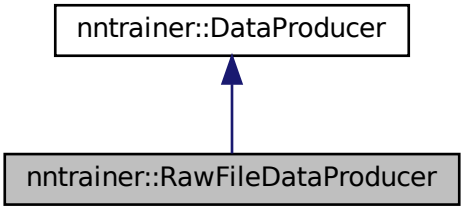
RawFileDataProducer which contains a callback and returns back.

`#include <raw_file_data_producer.h>`

Inheritance diagram for nntrainer::RawFileDataProducer:



Collaboration diagram for nntrainer::RawFileDataProducer:



Public Member Functions

- RawFileDataProducer ()

  Construct a new Raw File Data Producer object.
- RawFileDataProducer (const std::string &path)

  Construct a new RawFileDataProducer object.
- ∼RawFileDataProducer ()

  Destroy the RawFileDataProducer object.
- const std::string getType () const override

  Get the producer type.
- void setProperty (const std::vector< std::string > &properties) override
- DataProducer::Generator finalize (const std::vector< TensorDim > &input_dims, const std::vector< Tensor↩ Dim > &label_dims, void ∗user_data=nullptr) override
- unsigned int size (const std::vector< TensorDim > &input_dims, const std::vector< TensorDim > &label_↩ dims) const override
- void exportTo (Exporter &exporter, const ml::train::ExportMethods &method) const override

Static Public Attributes

- static constexpr unsigned int pixel_size
- static const std::string **type** = "file"

Additional Inherited Members

8.315.1 Detailed Description

RawFileDataProducer which contains a callback and returns back.

Definition at line 37 of file raw_file_data_producer.h.

8.315.2 Constructor & Destructor Documentation

8.315.2.1 RawFileDataProducer() [1/2]

```
nntrainer::RawFileDataProducer::RawFileDataProducer ( )
```

Construct a new Raw File Data Producer object.

Definition at line 29 of file raw_file_data_producer.cpp.

8.315.2.2 RawFileDataProducer() [2/2]

```
nntrainer::RawFileDataProducer::RawFileDataProducer (
            const std::string & path )
```

Construct a new RawFileDataProducer object.

Definition at line 31 of file raw_file_data_producer.cpp.

8.315.2.3 ~RawFileDataProducer()

```
nntrainer::RawFileDataProducer::~RawFileDataProducer ( )
```

Destroy the RawFileDataProducer object.

Definition at line 33 of file raw_file_data_producer.cpp.

8.315.3 Member Function Documentation

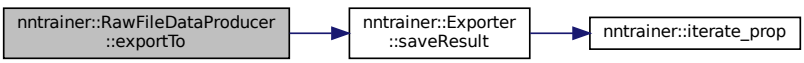
8.315.3.1 exportTo()

```
void nntrainer::RawFileDataProducer::exportTo (
            Exporter & exporter,
            const ml::train::ExportMethods & method ) const  [override], [virtual]
```

Reimplemented from nntrainer::DataProducer.

Definition at line 113 of file raw_file_data_producer.cpp.

Here is the call graph for this function:



8.315.3.2 finalize()

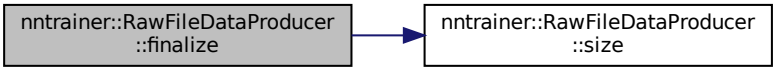
```
DataProducer::Generator nntrainer::RawFileDataProducer::finalize (
            const std::vector< TensorDim > & input_dims,
            const std::vector< TensorDim > & label_dims,
            void * user_data = nullptr )  [override], [virtual]
```

as we are passing the reference of file, this means created lamabda is tightly couple with the file, this is not desirable but working fine for now...

Reimplemented from nntrainer::DataProducer.

Definition at line 47 of file raw_file_data_producer.cpp.

Here is the call graph for this function:



8.315.3.3 getType()

```
const std::string nntrainer::RawFileDataProducer::getType ( ) const  [override], [virtual]
```

Get the producer type.

Returns

const std::string type representation

Implements nntrainer::DataProducer.

Definition at line 35 of file raw_file_data_producer.cpp.

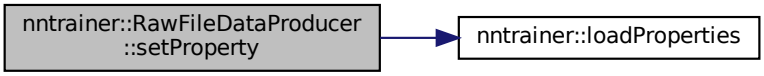
8.315.3.4 setProperty()

```
void nntrainer::RawFileDataProducer::setProperty (
            const std::vector< std::string > & properties )  [override], [virtual]
```

Reimplemented from nntrainer::DataProducer.

Definition at line 39 of file raw_file_data_producer.cpp.

Here is the call graph for this function:



8.315.3.5 size()

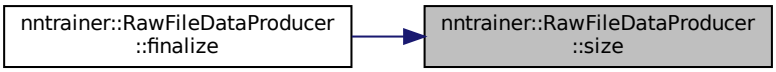
```
unsigned int nntrainer::RawFileDataProducer::size (
            const std::vector< TensorDim > & input_dims,
            const std::vector< TensorDim > & label_dims ) const  [override], [virtual]
```

checking alignment is a good way to make check if a file is valid, unfortunately, our test dataset does not have this property (trainingSet.dat, valSet.dat, testSet.dat) after checking, we can uncomment below line.

Reimplemented from nntrainer::DataProducer.

Definition at line 84 of file raw_file_data_producer.cpp.

Here is the caller graph for this function:



8.315.4 Member Data Documentation

8.315.4.1 pixel_size

```
constexpr unsigned int nntrainer::RawFileDataProducer::pixel_size  [inline], [static], [constexpr]
```

Initial value:

```
=
    sizeof(float)
```

Todo make this a configurable type

Definition at line 39 of file raw_file_data_producer.h.

The documentation for this class was generated from the following files:

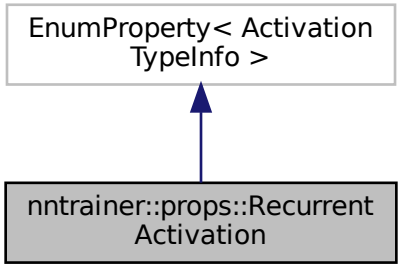
- dataset/raw_file_data_producer.h
- dataset/raw_file_data_producer.cpp

8.316 nntrainer::props::RecurrentActivation Class Reference

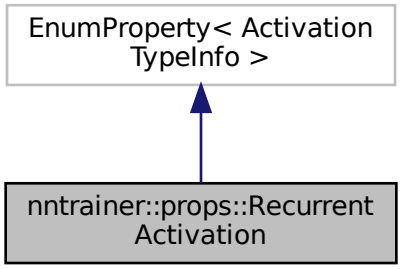
RecurrentActivation Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::RecurrentActivation:



Collaboration diagram for nntrainer::props::RecurrentActivation:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- RecurrentActivation (ActivationTypeInfo::Enum value=ActivationTypeInfo::Enum::ACT_NONE)

  Construct a new RecurrentActivation object with default value ActivationTypeInfo::Enum::ACT_NONE.

Static Public Attributes

- static constexpr const char * **key** = "recurrent_activation"

8.316.1 Detailed Description

RecurrentActivation Enumeration Information.

Definition at line 882 of file common_properties.h.

8.316.2 Constructor & Destructor Documentation

8.316.2.1 RecurrentActivation()

```
nntrainer::props::RecurrentActivation::RecurrentActivation (
            ActivationTypeInfo::Enum value = ActivationTypeInfo::Enum::ACT_NONE )
```

Construct a new RecurrentActivation object with default value ActivationTypeInfo::Enum::ACT_NONE.

Definition at line 293 of file common_properties.cpp.

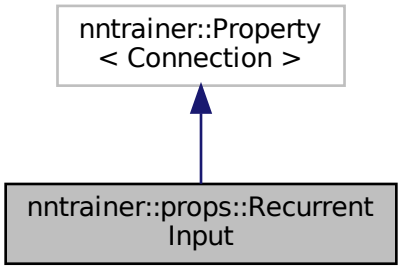
The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

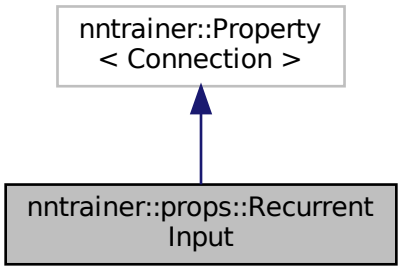
8.317 nntrainer::props::RecurrentInput Class Reference

Property for recurrent inputs.

Inheritance diagram for nntrainer::props::RecurrentInput:



Collaboration diagram for nntrainer::props::RecurrentInput:



Public Types

- using **prop_tag** = connection_prop_tag

Public Member Functions

- RecurrentInput ()

  Construct a new Recurrent Input object.
- RecurrentInput (const Connection &name)

  Construct a new Recurrent Input object.

Static Public Attributes

- static constexpr const char * **key** = "recurrent_input"

8.317.1 Detailed Description

Property for recurrent inputs.

Definition at line 68 of file recurrent_realizer.cpp.

8.317.2 Constructor & Destructor Documentation

8.317.2.1 RecurrentInput() [1/2]

```
nntrainer::props::RecurrentInput::RecurrentInput ( )
```

Construct a new Recurrent Input object.

Definition at line 86 of file recurrent_realizer.cpp.

8.317.2.2 RecurrentInput() [2/2]

```
nntrainer::props::RecurrentInput::RecurrentInput (
            const Connection & name )
```

Construct a new Recurrent Input object.

Parameters

name	name

Definition at line 87 of file recurrent_realizer.cpp.

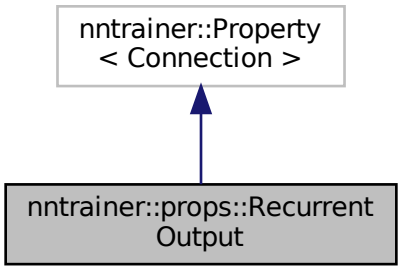
The documentation for this class was generated from the following file:

- compiler/recurrent_realizer.cpp

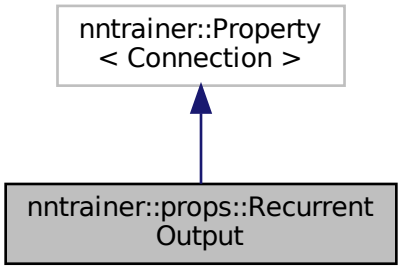
8.318 nntrainer::props::RecurrentOutput Class Reference

Property for recurrent outputs.

Inheritance diagram for nntrainer::props::RecurrentOutput:



Collaboration diagram for nntrainer::props::RecurrentOutput:



Public Types

- using **prop_tag** = connection_prop_tag

Public Member Functions

- RecurrentOutput ()

  Construct a new Recurrent Output object.
- RecurrentOutput (const Connection &name)

  Construct a new Recurrent Output object.

Static Public Attributes

- static constexpr const char * **key** = "recurrent_output"

8.318.1 Detailed Description

Property for recurrent outputs.

Definition at line 93 of file recurrent_realizer.cpp.

8.318.2 Constructor & Destructor Documentation

8.318.2.1 RecurrentOutput() [1/2]

```
nntrainer::props::RecurrentOutput::RecurrentOutput ( )
```

Construct a new Recurrent Output object.

Definition at line 111 of file recurrent_realizer.cpp.

8.318.2.2 RecurrentOutput() [2/2]

```
nntrainer::props::RecurrentOutput::RecurrentOutput (
            const Connection & name )
```

Construct a new Recurrent Output object.

Parameters

name	name

Definition at line 112 of file recurrent_realizer.cpp.

The documentation for this class was generated from the following file:

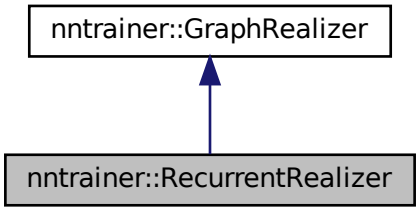
- compiler/recurrent_realizer.cpp

8.319 nntrainer::RecurrentRealizer Class Reference

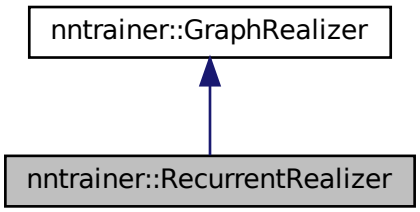
Recurrent Realizer which unrolls graph from given graph representation.

`#include <recurrent_realizer.h>`

Inheritance diagram for nntrainer::RecurrentRealizer:



Collaboration diagram for nntrainer::RecurrentRealizer:



Public Member Functions

- RecurrentRealizer (const std::vector< std::string > &properties, const std::vector< Connection > &input_↩ conns, const std::vector< Connection > &end_conns)

  Construct a new Recurrent Realizer object.
- RecurrentRealizer (const char *ini, const std::vector< std::string > &external_input_layers)

  Construct a new Recurrent Realizer object.
- ~RecurrentRealizer ()

  Destroy the Recurrent Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  realized graph

8.319.1 Detailed Description

Recurrent Realizer which unrolls graph from given graph representation.

Definition at line 44 of file recurrent_realizer.h.

8.319.2 Constructor & Destructor Documentation

8.319.2.1 RecurrentRealizer() [1/2]

```
nntrainer::RecurrentRealizer::RecurrentRealizer (
            const std::vector< std::string > & properties,
            const std::vector< Connection > & input_conns,
            const std::vector< Connection > & end_conns )
```

Construct a new Recurrent Realizer object.

Note

There are three types of input_layers in recurrent realizer

1. input_layers: input layers
2. external_input_layers: input_layers being renamed to
3. recurrent_input: Override it's input layers to recurrent output for the steps, where steps > 0

Parameters

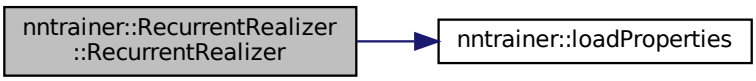
properties	unroll_for = <int> // define timestep of unrolling return_sequences = <bool> // return sequences recurrent_inputs = <vector<std::string>> // start of the loop recurrent_ouptuts = <vector<std::string>> // end of the loop
input_conns	input conns from outer side
end_conns	end connections (output of the internal graph)

build end info. eg) end_layers: a(0), a(3), b(0) becomes end_info: {{a, 3}, {b, 0}} end_layers: a(1), b(3), c(0) becomes end_info: {{a, 1}, {b, 3}, {c, 0}}

Todo Deal as sequence as proper connection with identity layer

Definition at line 115 of file recurrent_realizer.cpp.

Here is the call graph for this function:



8.319.2.2 RecurrentRealizer() [2/2]

```
nntrainer::RecurrentRealizer::RecurrentRealizer (
            const char * ini,
            const std::vector< std::string > & external_input_layers )
```

Construct a new Recurrent Realizer object.

Parameters

ini	ini to load recurrent properties from
external_input_layers	external input layers to map input layers

Todo delegate to RecurrentRealizer(

NYI!

Definition at line 214 of file recurrent_realizer.cpp.

8.319.2.3 ~RecurrentRealizer()

```
nntrainer::RecurrentRealizer::~RecurrentRealizer ( )
```

Destroy the Recurrent Realizer object.

Definition at line 222 of file recurrent_realizer.cpp.

8.319.3 Member Function Documentation

8.319.3.1 realize()

```
GraphRepresentation nntrainer::RecurrentRealizer::realize (
            const GraphRepresentation & reference )  [override], [virtual]
```

realized graph

Parameters

reference	reference to realize graph

Returns

GraphRepresentation realized graph

empty intended

maps input place holder to given name otherwise scopped to suffix "/0"

#1744, quick fix, add shared_from to every node

Create a single time step. Used inside step2_unroll.

1. remap identifiers to $name/$idx
2. override first output name to $name/$idx - 1
3. set shared_from
4. if recurrent layer type set timestep property

unroll the graph by calling create_step()

case when return sequence is true, concat layer is added to aggregate all the output

Todo have axis in concat layer

Todo this has to be wrapped with identity layer as #1793

create identity layer with output name by either creating concat layer or directly using the connection, the number of inputs connection have is depending on the end_conns max.

eg) layer A outputs a, b, c, d

if end_layers=A(0),A(2) as_sequence=A(0) realizer cannot know there is d so this is ignored. It is okay because user didn't specify to use it anyway

[A] type=identity input_layers=A_concat_0, A(1), A(2)

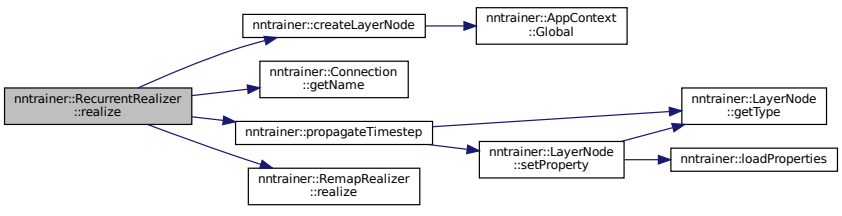
Note

below is inefficient way of processing nodes. consider optimize below as needed by calling remap realizer only once

Implements nntrainer::GraphRealizer.

Definition at line 225 of file recurrent_realizer.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

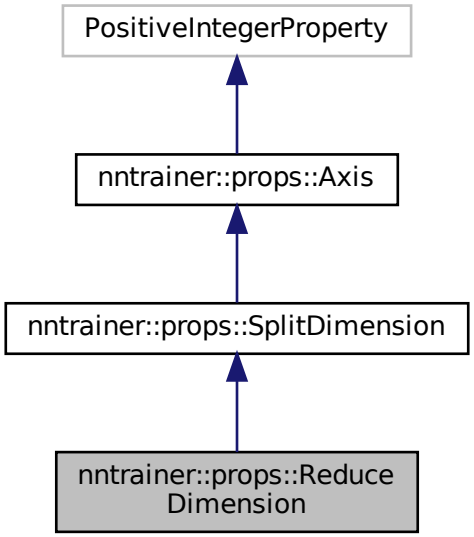
- compiler/recurrent_realizer.h
- compiler/recurrent_realizer.cpp

8.320 nntrainer::props::ReduceDimension Class Reference

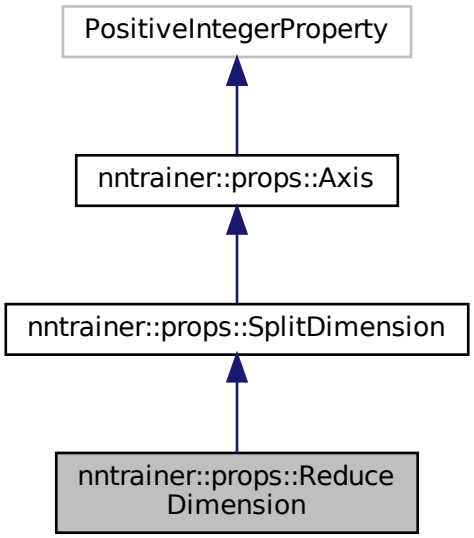
ReduceDimension property, dimension along which to reduce the input.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::ReduceDimension:



Collaboration diagram for nntrainer::props::ReduceDimension:



Additional Inherited Members

8.320.1 Detailed Description

ReduceDimension property, dimension along which to reduce the input.

Definition at line 314 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.321 tflite::ReducerOptionsBuilder Struct Reference

Public Types

- typedef ReducerOptions **Table**

Public Member Functions

- void **add_keep_dims** (bool keep_dims)
- **ReducerOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ReducerOptionsBuilder & **operator=** (const ReducerOptionsBuilder &)
- flatbuffers::Offset< ReducerOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.321.1 Detailed Description

Definition at line 4737 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.322 nntrainer::props::RegularizerInfo Struct Reference

Enumeration of tensor regularization type.

`#include <common_properties.h>`

Public Types

- using **Enum** = nntrainer::WeightRegularizer

Static Public Attributes

- static constexpr std::initializer_list< Enum > **EnumList**
- static constexpr const char * **EnumStr** [ ] = {"l2norm", "none", "unknown"}

8.322.1 Detailed Description

Enumeration of tensor regularization type.

Definition at line 999 of file common_properties.h.

8.322.2 Member Data Documentation

8.322.2.1 EnumList

```
constexpr std::initializer_list<Enum> nntrainer::props::RegularizerInfo::EnumList  [static],
[constexpr]
```

Initial value:

```
= {
    Enum::L2NORM, Enum::NONE, Enum::UNKNOWN}
```

Definition at line 1001 of file common_properties.h.

The documentation for this struct was generated from the following file:

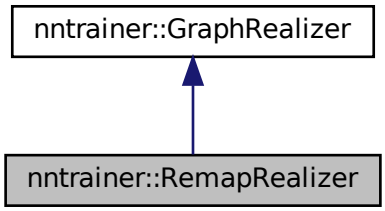
- layers/common_properties.h

8.323 nntrainer::RemapRealizer Class Reference

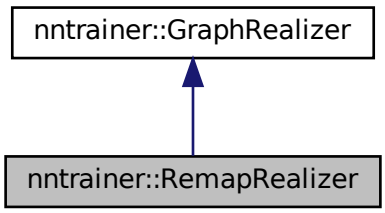
Graph realizer class which remaps identifiers inside the graph representation, remap_function will be applied for all the layers identifier visible.

`#include <remap_realizer.h>`

Inheritance diagram for nntrainer::RemapRealizer:



Collaboration diagram for nntrainer::RemapRealizer:



Public Member Functions

- RemapRealizer (std::function< void(std::string &, unsigned &)> remap_connection_function)

  Construct a new Remap Realizer object (connection mode)
- RemapRealizer (std::function< void(std::string &)> remap_function)

  Construct a new Remap Realizer object (identifier mode)
- ~RemapRealizer ()

  Destroy the Graph Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  graph realizer creates a new graph based on the reference

8.323.1 Detailed Description

Graph realizer class which remaps identifiers inside the graph representation, remap_function will be applied for all the layers identifier visible.

Definition at line 29 of file remap_realizer.h.

8.323.2 Constructor & Destructor Documentation

8.323.2.1 RemapRealizer() [1/2]

```
nntrainer::RemapRealizer::RemapRealizer (
            std::function< void(std::string &, unsigned &)> remap_connection_function )
```

Construct a new Remap Realizer object (connection mode)

Parameters

remap_connection_function	remap connection function, with this constructor, only connections will be remapped eg) if you are inserting a new layer node in between, only connections need remapping

Definition at line 17 of file remap_realizer.cpp.

8.323.2.2 RemapRealizer() [2/2]

```
nntrainer::RemapRealizer::RemapRealizer (
            std::function< void(std::string &)> remap_function )
```

Construct a new Remap Realizer object (identifier mode)

Parameters

remap_function	remap function, with this constructor, all identifiers wherever it is used will be remapped

Definition at line 26 of file remap_realizer.cpp.

8.323.2.3 ~RemapRealizer()

```
nntrainer::RemapRealizer::~RemapRealizer ( )
```

Destroy the Graph Realizer object.

Definition at line 35 of file remap_realizer.cpp.

8.323.3 Member Function Documentation

8.323.3.1 realize()

```
GraphRepresentation nntrainer::RemapRealizer::realize (
            const GraphRepresentation & reference )  [override], [virtual]
```

graph realizer creates a new graph based on the reference

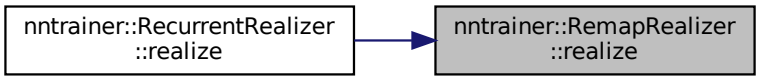
Note

while remap realization, the graph is invalid.

Implements nntrainer::GraphRealizer.

Definition at line 38 of file remap_realizer.cpp.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

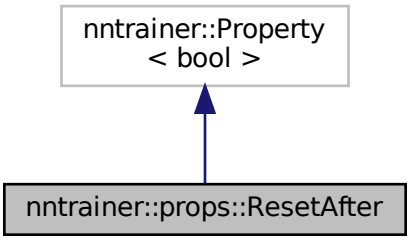
- compiler/remap_realizer.h
- compiler/remap_realizer.cpp

8.324 nntrainer::props::ResetAfter Class Reference

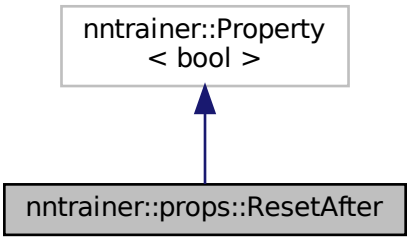
ResetAfter property, apply reset gate after matrix multiplication if this property is true. Apply before the multiplication if false. Used in gru, grucell.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::ResetAfter:



Collaboration diagram for nntrainer::props::ResetAfter:



Public Types

- using prop_tag = bool_prop_tag

Public Member Functions

- ResetAfter (bool value=true)

  Construct a new ResetAfter object with a default value true.

Static Public Attributes

- static constexpr const char * key = "reset_after"

8.324.1 Detailed Description

ResetAfter property, apply reset gate after matrix multiplication if this property is true. Apply before the multiplication if false. Used in gru, grucell.

Definition at line 687 of file common_properties.h.

8.324.2 Member Typedef Documentation

8.324.2.1 prop_tag

```
using nntrainer::props::ResetAfter::prop_tag = bool_prop_tag
```

property type

Definition at line 696 of file common_properties.h.

8.324.3 Constructor & Destructor Documentation

8.324.3.1 ResetAfter()

```
nntrainer::props::ResetAfter::ResetAfter (
            bool value = true )  [inline]
```

Construct a new ResetAfter object with a default value true.

Definition at line 694 of file common_properties.h.

8.324.4 Member Data Documentation

8.324.4.1 key

```
constexpr const char* nntrainer::props::ResetAfter::key = "reset_after"  [static], [constexpr]
```

unique key to access

Definition at line 695 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.325 tflite::ReshapeOptionsBuilder Struct Reference

Public Types

- typedef ReshapeOptions **Table**

Public Member Functions

- void **add_new_shape** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> new_shape)
- **ReshapeOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ReshapeOptionsBuilder & **operator=** (const ReshapeOptionsBuilder &)
- flatbuffers::Offset< ReshapeOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.325.1 Detailed Description

Definition at line 4182 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.326 tflite::ResizeBilinearOptionsBuilder Struct Reference

Public Types

- typedef ResizeBilinearOptions **Table**

Public Member Functions

- void **add_align_corners** (bool align_corners)
- void **add_half_pixel_centers** (bool half_pixel_centers)
- **ResizeBilinearOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ResizeBilinearOptionsBuilder & **operator=** (const ResizeBilinearOptionsBuilder &)
- flatbuffers::Offset< ResizeBilinearOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.326.1 Detailed Description

Definition at line 3980 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.327 tflite::ResizeNearestNeighborOptionsBuilder Struct Reference

Public Types

- typedef ResizeNearestNeighborOptions **Table**

Public Member Functions

- void **add_align_corners** (bool align_corners)
- void **add_half_pixel_centers** (bool half_pixel_centers)
- **ResizeNearestNeighborOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ResizeNearestNeighborOptionsBuilder & **operator=** (const ResizeNearestNeighborOptionsBuilder &)
- flatbuffers::Offset< ResizeNearestNeighborOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.327.1 Detailed Description

Definition at line 4032 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

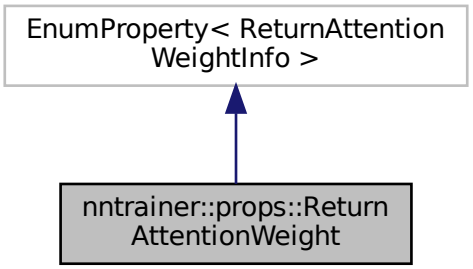
- compiler/tf_schema_generated.h

8.328 nntrainer::props::ReturnAttentionWeight Class Reference

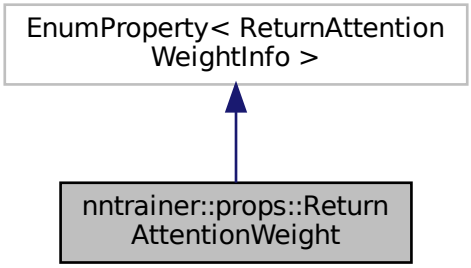
ReturnAttentionWeight, return attention weight.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::ReturnAttentionWeight:



Collaboration diagram for nntrainer::props::ReturnAttentionWeight:



Public Types

- using prop_tag = enum_class_prop_tag

Public Member Functions

- ReturnAttentionWeight (ReturnAttentionWeightInfo::Enum value=ReturnAttentionWeightInfo::Enum::none)

  Construct a new ReturnAttentionWeight object.

Static Public Attributes

- static constexpr const char * key

8.328.1 Detailed Description

ReturnAttentionWeight, return attention weight.

"none" won't return attention weight. "before"/"after" will return attention weight before/after applying dropout

Note

Correspond with return_attention_scores of tensorflow and Correspond with need_weights of torch

Definition at line 1243 of file common_properties.h.

8.328.2 Member Typedef Documentation

8.328.2.1 prop_tag

```
using nntrainer::props::ReturnAttentionWeight::prop_tag = enum_class_prop_tag
```

property type

Definition at line 1247 of file common_properties.h.

8.328.3 Constructor & Destructor Documentation

8.328.3.1 ReturnAttentionWeight()

```
nntrainer::props::ReturnAttentionWeight::ReturnAttentionWeight (
            ReturnAttentionWeightInfo::Enum value = ReturnAttentionWeightInfo::Enum::none )
```

Construct a new ReturnAttentionWeight object.

Definition at line 341 of file common_properties.cpp.

8.328.4 Member Data Documentation

8.328.4.1 key

```
constexpr const char* nntrainer::props::ReturnAttentionWeight::key  [static], [constexpr]
```

Initial value:

```
=
    "return_attention_weight"
```

unique key to access

Definition at line 1245 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.329 nntrainer::props::ReturnAttentionWeightInfo Struct Reference

Enumeration of return attention weight.

```
#include <common_properties.h>
```

Public Types

- enum **Enum** { **none**, **before**, **after** }

Static Public Attributes

- static constexpr std::initializer_list< Enum > **EnumList**
- static constexpr const char * **EnumStr** [ ] = {"none", "before", "after"}

8.329.1 Detailed Description

Enumeration of return attention weight.

Definition at line 1226 of file common_properties.h.

8.329.2 Member Data Documentation

8.329.2.1 EnumList

```
constexpr std::initializer_list<Enum> nntrainer::props::ReturnAttentionWeightInfo::EnumList
[static], [constexpr]
```

Initial value:

```
= {
    Enum::none, Enum::before, Enum::after}
```

Definition at line 1228 of file common_properties.h.

The documentation for this struct was generated from the following file:

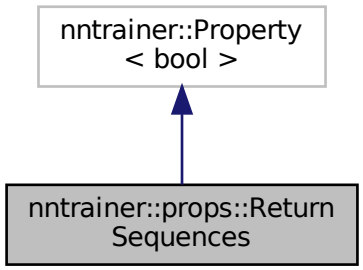
- layers/common_properties.h

8.330 nntrainer::props::ReturnSequences Class Reference

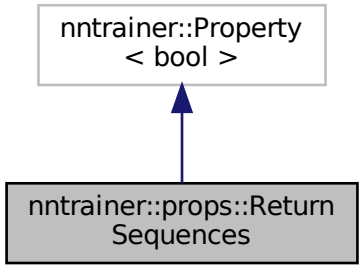
return sequence property, used to check whether return only the last output. Return last output if true.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::ReturnSequences:



Collaboration diagram for nntrainer::props::ReturnSequences:



Public Types

- using **prop_tag** = bool_prop_tag

Public Member Functions

- ReturnSequences (bool value=false)

  Construct a new ReturnSequences object.

Static Public Attributes

- static constexpr const char * **key** = "return_sequences"

8.330.1 Detailed Description

return sequence property, used to check whether return only the last output. Return last output if true.

Definition at line 633 of file common_properties.h.

8.330.2 Constructor & Destructor Documentation

8.330.2.1 ReturnSequences()

```
nntrainer::props::ReturnSequences::ReturnSequences (
            bool value = false )
```

Construct a new ReturnSequences object.

Definition at line 79 of file common_properties.cpp.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.331 tflite::ReverseSequenceOptionsBuilder Struct Reference

Public Types

- typedef ReverseSequenceOptions **Table**

Public Member Functions

- void **add_seq_dim** (int32_t seq_dim)
- void **add_batch_dim** (int32_t batch_dim)
- **ReverseSequenceOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ReverseSequenceOptionsBuilder & **operator=** (const ReverseSequenceOptionsBuilder &)
- flatbuffers::Offset< ReverseSequenceOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.331.1 Detailed Description

Definition at line 6588 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.332 tflite::ReverseV2OptionsBuilder Struct Reference

Public Types

- typedef ReverseV2Options **Table**

Public Member Functions

- **ReverseV2OptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ReverseV2OptionsBuilder & **operator=** (const ReverseV2OptionsBuilder &)
- flatbuffers::Offset< ReverseV2Options > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.332.1 Detailed Description

Definition at line 6456 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.333 tflite::RNNOptionsBuilder Struct Reference

Public Types

- typedef RNNOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_asymmetric_quantize_inputs** (bool asymmetric_quantize_inputs)
- **RNNOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- RNNOptionsBuilder & **operator=** (const RNNOptionsBuilder &)
- flatbuffers::Offset< RNNOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.333.1 Detailed Description

Definition at line 3164 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.334 nntrainer::RunLayerContext Class Reference

Public Member Functions

- RunLayerContext ()

  Construct a new Run Layer Context object.
- RunLayerContext (const std::string &name, bool in_place_)

  Construct a new Run Layer Context object.
- RunLayerContext (const std::string &name, bool trainable, float l, bool in_place_, const std::vector< Weight * > &w, const std::vector< Var_Grad * > &in, const std::vector< Var_Grad * > &out, const std::vector< Var_Grad * > &t)

  Construct a new Run Layer Context object.
- Tensor & getWeight (unsigned int idx) const

  Get the Weight tensor object.
- Tensor & getWeightGrad (unsigned int idx) const

  Get the Weight Gradient tensor object.
- Tensor & getWeightOptVar (unsigned int idx, unsigned int jdx) const

  Get the Weight Optimizer Variable tensor object.
- const std::string & getWeightName (unsigned int idx) const

  Get the Weight name.
- bool weightHasGradient (unsigned int idx) const

  check if the weight has gradient
- Tensor & getOutput (unsigned int idx)

  Get the Output tensor object.
- const Tensor & getOutput (unsigned int idx) const

  Get the Output tensor object.
- const Tensor getOutputGrad (unsigned int idx) const

  Get the Output Grad tensor object.
- Tensor & getOutputGradUnsafe (unsigned int idx)

  Get the Output Grad tensor object.
- bool outputHasGradient (unsigned int idx) const

  check if the weight has gradient
- const Tensor getIncomingDerivative (unsigned int idx) const

  Get the incoming Derivative tensor object.
- Tensor & getInput (unsigned int idx)

Get the Input tensor object.

- const Tensor & getInput (unsigned int idx) const

  Get the Input tensor object.
- Tensor & getInputGrad (unsigned int idx)

  Get the Input Grad tensor object.
- bool inputHasGradient (unsigned int idx) const

  check if the weight has gradient
- Tensor & getOutgoingDerivative (unsigned int idx)

  Get the outgoing Derivative tensor object.
- Tensor & getTensor (unsigned int idx)

  Get the Tensor object.
- const Tensor & getTensor (unsigned int idx) const

  Get the Tensor object.
- Tensor & getTensorGrad (unsigned int idx)

  Get the Tensor Grad object.
- const Tensor & getTensorGrad (unsigned int idx) const

  Get the Tensor Grad object.
- bool tensorHasGradient (unsigned int idx) const

  check if the tensor has gradient
- bool isWeightDependent (unsigned int idx) const

  check if the weight is burrowed from others so it is dependent
- bool isGradientFirstAccess (unsigned int idx) const

  check current gradient is first access
- bool isGradientLastAccess (unsigned int idx) const

  check current gradient is last access
- bool isGradientClipByGlobalNorm (unsigned int idx) const

  check if the gradient is to be clipped by global norm
- const std::string & getTensorName (unsigned int idx) const

  Get the tensor name.
- unsigned int getNumOutputs () const

  Get the number of Outputs tensor objects.
- unsigned int getNumInputs () const

  Get the number of inputs tensor objects.
- unsigned int getNumWeights () const

  Get the number of weights tensor objects.
- unsigned int getNumWeightOptVar (unsigned int idx) const

  Get the Number of Weight Optimizer Variable tensor object.
- unsigned int getNumTensors () const

  Get the number of requested tensors objects.
- void setBatch (unsigned int batch)

  Set the batch for the run context.
- void updateTensor (unsigned int idx, unsigned int batch)

  Update the dimensions for a requested tensor.
- Weight & getWeightObject (unsigned int idx)

  Get weight object for the weights.
- bool isLabelAvailable (unsigned int idx) const

  check if the label is available
- Tensor & getLabel (unsigned int idx)

  Get label tensor.
- void setLoss (float val)

  update loss by the layer

- float getLoss () const
  update loss by the layer
- float getRegularizationLoss () const
  get regularization loss of the layer
- const std::string & getName () const
  get name by the layer
- bool getTrainable () const
  get trainable by the layer
- bool readyToUse () const
  check if run context is set and is ready to use
- bool validate (bool skip_input=false, bool skip_label=false)
  validates the run context after run
- bool executeInPlace () const
  check if the layer is expected to run in-place

8.334.1 Detailed Description

Definition at line 315 of file layer_context.h.

8.334.2 Constructor & Destructor Documentation

8.334.2.1 RunLayerContext() [1/3]

```
nntrainer::RunLayerContext::RunLayerContext ( )  [inline]
```

Construct a new Run Layer Context object.

Definition at line 321 of file layer_context.h.

8.334.2.2 RunLayerContext() [2/3]

```
nntrainer::RunLayerContext::RunLayerContext (
            const std::string & name,
            bool in_place_ )  [inline]
```

Construct a new Run Layer Context object.

Definition at line 327 of file layer_context.h.

8.334.2.3 RunLayerContext() [3/3]

```
nntrainer::RunLayerContext::RunLayerContext (
            const std::string & name,
            bool trainable,
            float l,
            bool in_place_,
            const std::vector< Weight * > & w,
            const std::vector< Var_Grad * > & in,
            const std::vector< Var_Grad * > & out,
            const std::vector< Var_Grad * > & t )
```

Construct a new Run Layer Context object.

Parameters

name	name of the layer
trainable	if the layer is trainable
l	loss of the layer
*in_↩ place_*	execution in-place of the layer
w	weights of the layer
in	inputs of the layer
out	outputs of the layer
t	extra tensors of the layer

Definition at line 122 of file layer_context.cpp.

8.334.3 Member Function Documentation

8.334.3.1 executeInPlace()

```
bool nntrainer::RunLayerContext::executeInPlace ( ) const  [inline]
```

check if the layer is expected to run in-place

Returns

true if in-place, else false

Definition at line 713 of file layer_context.h.

8.334.3.2 getIncomingDerivative()

```
const Tensor nntrainer::RunLayerContext::getIncomingDerivative (
            unsigned int idx ) const
```

Get the incoming Derivative tensor object.

Parameters

idx	Identifier of the output

Returns

Tensor output derivative tensor, if derivative does not have gradient, return a temporary, initialized to zero

Parameters

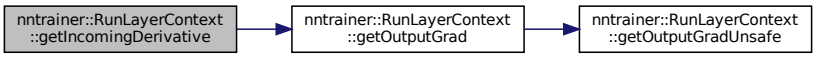
idx	Identifier of the output

Returns

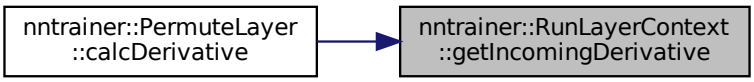
Tensor tensor to incoming derivative. If

Definition at line 274 of file layer_context.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.334.3.3 getInput() [1/2]

```
Tensor & nntrainer::RunLayerContext::getInput (
            unsigned int idx )
```

Get the Input tensor object.

Parameters

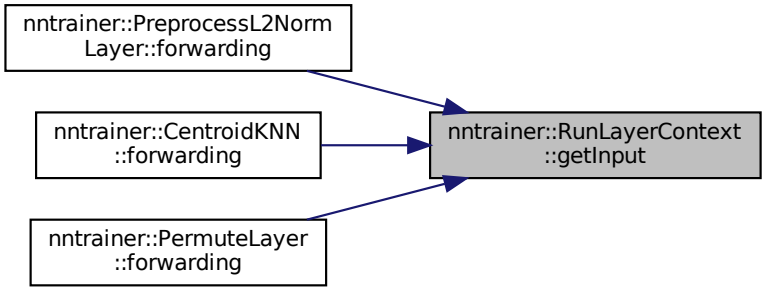
idx	Identifier of the input

Returns

Tensor& Reference to the input grad tensor

Definition at line 284 of file layer_context.cpp.

Here is the caller graph for this function:



8.334.3.4 getInput() [2/2]

```
const Tensor & nntrainer::RunLayerContext::getInput (
            unsigned int idx ) const
```

Get the Input tensor object.

Parameters

idx	Identifier of the input

Returns

Tensor& Reference to the input grad tensor

Definition at line 288 of file layer_context.cpp.

8.334.3.5 getInputGrad()

```
Tensor & nntrainer::RunLayerContext::getInputGrad (
            unsigned int idx )
```

Get the Input Grad tensor object.

Parameters

idx	Identifier of the input

Returns

Tensor& Reference to the input grad tensor

Definition at line 298 of file layer_context.cpp.

Here is the caller graph for this function:

8.334.3.6 getLabel()

```
Tensor & nntrainer::RunLayerContext::getLabel (
            unsigned int idx )
```

Get label tensor.

Parameters

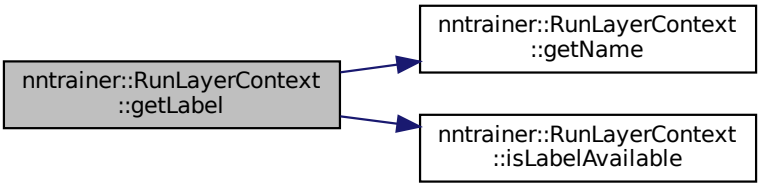
idx	Identifier of the input

Returns

Tensor& Reference to the label tensor

Definition at line 457 of file layer_context.cpp.

Here is the call graph for this function:



8.334.3.7 getLoss()

```
float nntrainer::RunLayerContext::getLoss ( ) const  [inline]
```

update loss by the layer

Returns

loss of the layer

Note

does not includes the regularization loss.

Definition at line 662 of file layer_context.h.

8.334.3.8 getName()

```
const std::string& nntrainer::RunLayerContext::getName ( ) const  [inline]
```

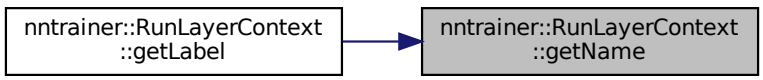
get name by the layer

Returns

name of the layer

Definition at line 682 of file layer_context.h.

Here is the caller graph for this function:



8.334.3.9 getNumInputs()

```
unsigned int nntrainer::RunLayerContext::getNumInputs ( ) const  [inline]
```

Get the number of inputs tensor objects.

Returns

unsigned int number of input tensors

Definition at line 584 of file layer_context.h.

8.334.3.10 getNumOutputs()

```
unsigned int nntrainer::RunLayerContext::getNumOutputs ( ) const  [inline]
```

Get the number of Outputs tensor objects.

Returns

unsigned int number of output tensors

Definition at line 577 of file layer_context.h.

8.334.3.11 getNumTensors()

```
unsigned int nntrainer::RunLayerContext::getNumTensors ( ) const  [inline]
```

Get the number of requested tensors objects.

Returns

unsigned int number of requested tensors

Definition at line 606 of file layer_context.h.

8.334.3.12 getNumWeightOptVar()

```
unsigned int nntrainer::RunLayerContext::getNumWeightOptVar (
            unsigned int idx ) const
```

Get the Number of Weight Optimizer Variable tensor object.

Parameters

idx	Identifier of the weight

Returns

unsigned int Number of the weight optimizer variable

Parameters

idx	Identifier of the weight

Returns

int Number of the weight optimizer variable

Definition at line 184 of file layer_context.cpp.

8.334.3.13 getNumWeights()

```
unsigned int nntrainer::RunLayerContext::getNumWeights ( ) const  [inline]
```

Get the number of weights tensor objects.

Returns

unsigned int number of weight tensors

Definition at line 591 of file layer_context.h.

Here is the caller graph for this function:

8.334.3.14 getOutgoingDerivative()

```
Tensor & nntrainer::RunLayerContext::getOutgoingDerivative (
            unsigned int idx )
```

Get the outgoing Derivative tensor object.

Parameters

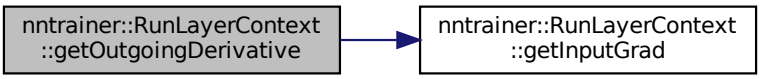
idx	Identifier of the input

Returns

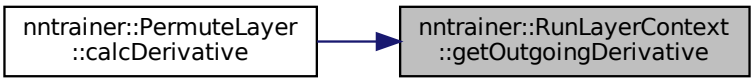
Tensor& Reference to the input derivative tensor

Definition at line 323 of file layer_context.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.334.3.15 getOutput() [1/2]

```
Tensor & nntrainer::RunLayerContext::getOutput (
             unsigned int idx )
```

Get the Output tensor object.

Parameters

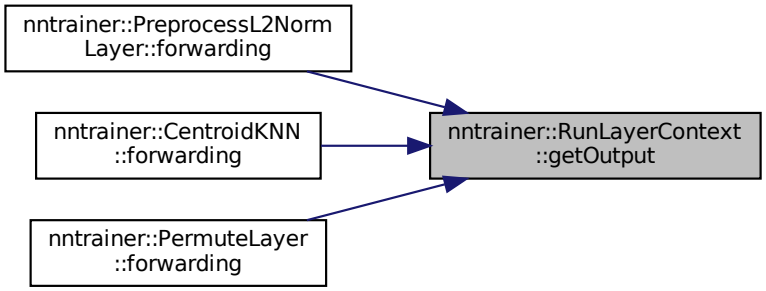
idx	Identifier of the output

Returns

Tensor& Reference to the output tensor

Definition at line 224 of file layer_context.cpp.

Here is the caller graph for this function:



8.334.3.16 getOutput() [2/2]

```
const Tensor & nntrainer::RunLayerContext::getOutput (
             unsigned int idx ) const
```

Get the Output tensor object.

Parameters

idx	Identifier of the output

Returns

Tensor& Reference to the output tensor

Definition at line 228 of file layer_context.cpp.

8.334.3.17 getOutputGrad()

```
const Tensor nntrainer::RunLayerContext::getOutputGrad (
             unsigned int idx ) const
```

Get the Output Grad tensor object.

Parameters

idx	Identifier of the output

Returns

Read-only output grad tensor, if derivative does not have gradient, return a temporary, initialized to zero

Parameters

idx	Identifier of the output

Returns

Tensor Read-only output grad tensor

Definition at line 238 of file layer_context.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

8.334.3.18 getOutputGradUnsafe()

```
Tensor & nntrainer::RunLayerContext::getOutputGradUnsafe (
            unsigned int idx )
```

Get the Output Grad tensor object.

Parameters

idx	Identifier of the output

Returns

Tensor& Reference to the output grad tensor, this is valid only if the given output is trainable

Note

recommended to NOT use this function as a layer developer but rather use getOutputGrad().

Parameters

idx	Identifier of the output

Returns

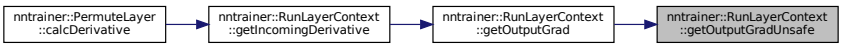
Tensor& Reference to the output grad tensor

Note

recommended to NOT use this function as a layer developer but rather use getOutputGrad().

Definition at line 264 of file layer_context.cpp.

Here is the caller graph for this function:



8.334.3.19 getRegularizationLoss()

```
float nntrainer::RunLayerContext::getRegularizationLoss ( ) const  [inline]
```

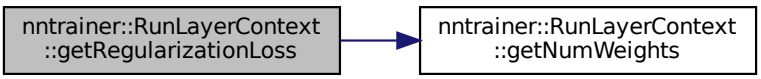
get regularization loss of the layer

Returns

regularization loss of the layer

Definition at line 669 of file layer_context.h.

Here is the call graph for this function:



8.334.3.20 getTensor() [1/2]

```
Tensor & nntrainer::RunLayerContext::getTensor (
             unsigned int idx )
```

Get the Tensor object.

Parameters

idx	Identifier of the tensor

Returns

Tensor& Reference to the tensor

Definition at line 333 of file layer_context.cpp.

8.334.3.21 getTensor() [2/2]

```
const Tensor & nntrainer::RunLayerContext::getTensor (
             unsigned int idx ) const
```

Get the Tensor object.

Parameters

idx	Identifier of the tensor

Returns

Tensor& Reference to the tensor

Definition at line 343 of file layer_context.cpp.

8.334.3.22 getTensorGrad() [1/2]

```
Tensor & nntrainer::RunLayerContext::getTensorGrad (
             unsigned int idx )
```

Get the Tensor Grad object.

Parameters

idx	Identifier of the tensor

Returns

Tensor& Reference to the tensor grad tensor

Definition at line 353 of file layer_context.cpp.

8.334.3.23 getTensorGrad() [2/2]

```
const Tensor & nntrainer::RunLayerContext::getTensorGrad (
             unsigned int idx ) const
```

Get the Tensor Grad object.

Parameters

idx	Identifier of the tensor

Returns

Tensor& Reference to the tensor grad tensor

Definition at line 366 of file layer_context.cpp.

8.334.3.24 getTensorName()

```
const std::string & nntrainer::RunLayerContext::getTensorName (
             unsigned int idx ) const
```

Get the tensor name.

Parameters

idx	Identifier of the tensor

Returns

name of the tensor

Definition at line 405 of file layer_context.cpp.

8.334.3.25 getTrainable()

```
bool nntrainer::RunLayerContext::getTrainable ( ) const  [inline]
```

get trainable by the layer

Returns

trainable of the layer

Definition at line 689 of file layer_context.h.

8.334.3.26 getWeight()

```
Tensor & nntrainer::RunLayerContext::getWeight (
            unsigned int idx ) const
```

Get the Weight tensor object.

Parameters

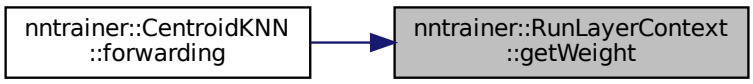
idx	Identifier of the weight

Returns

Tensor& Reference to the weight tensor

Definition at line 149 of file layer_context.cpp.

Here is the caller graph for this function:



8.334.3.27 getWeightGrad()

```
Tensor & nntrainer::RunLayerContext::getWeightGrad (
            unsigned int idx ) const
```

Get the Weight Gradient tensor object.

Note

this method returns the fresh gradient to be filled

Parameters

idx	Identifier of the weight

Returns

Tensor& Reference to the weight grad tensor

Parameters

idx	Identifier of the weight

Returns

Tensor& Reference to the weight grad tensor

Definition at line 159 of file layer_context.cpp.

8.334.3.28 getWeightName()

```
const std::string & nntrainer::RunLayerContext::getWeightName (
            unsigned int idx ) const
```

Get the Weight name.

Parameters

idx	Identifier of the weight

Returns

name of the weight

Definition at line 204 of file layer_context.cpp.

8.334.3.29 getWeightObject()

```
Weight & nntrainer::RunLayerContext::getWeightObject (
            unsigned int idx )
```

Get weight object for the weights.

Parameters

idx	index of the weight (identifier)

Returns

weight object

Definition at line 437 of file layer_context.cpp.

8.334.3.30 getWeightOptVar()

```
Tensor & nntrainer::RunLayerContext::getWeightOptVar (
            unsigned int idx,
            unsigned int jdx ) const
```

Get the Weight Optimizer Variable tensor object.

Parameters

idx	Identifier of the weight
jdx	Identifier of the weight optimizer variable

Returns

Tensor& Reference to the weight grad tensor

Parameters

idx	Identifier of the weight
jdx	Identifier of the optimizer variables

Returns

Tensor& Reference to the weight optimizer variable tensor

Definition at line 173 of file layer_context.cpp.

8.334.3.31 inputHasGradient()

```
bool nntrainer::RunLayerContext::inputHasGradient (
            unsigned int idx ) const
```

check if the weight has gradient

check if the input has gradient

Parameters

idx	Identifier of the weight

Returns

true if weight has gradient, else false

Parameters

idx	Identifier of the input

Returns

true if output has gradient, else false

Definition at line 313 of file layer_context.cpp.

8.334.3.32 isGradientClipByGlobalNorm()

```
bool nntrainer::RunLayerContext::isGradientClipByGlobalNorm (
            unsigned int idx ) const
```

check if the gradient is to be clipped by global norm

Parameters

idx	index

Returns

bool true if it is to be clipped else false

Definition at line 395 of file layer_context.cpp.

8.334.3.33 isGradientFirstAccess()

```
bool nntrainer::RunLayerContext::isGradientFirstAccess (
            unsigned int idx ) const
```

check current gradient is first access

Note

for now, it equivalent to weight last access, so this value is accessible for non-trainable weights as well. This is in terms of execution order.

Parameters

idx	index

Returns

bool true if first access

Definition at line 387 of file layer_context.cpp.

8.334.3.34 isGradientLastAccess()

```
bool nntrainer::RunLayerContext::isGradientLastAccess (
            unsigned int idx ) const
```

check current gradient is last access

Note

for now, it equivalent to weight last access, so this value is accessible for non-trainable weights as well. This is in terms of execution order.

Parameters

idx	index

Returns

bool true if last access

Definition at line 391 of file layer_context.cpp.

8.334.3.35 isLabelAvailable()

```
bool nntrainer::RunLayerContext::isLabelAvailable (
            unsigned int idx ) const
```

check if the label is available

Parameters

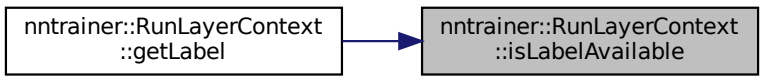
idx	Identifier of the input

Returns

true if label is available else false

Definition at line 447 of file layer_context.cpp.

Here is the caller graph for this function:



8.334.3.36 isWeightDependent()

```
bool nntrainer::RunLayerContext::isWeightDependent (
            unsigned int idx ) const
```

check if the weight is burrowed from others so it is dependent

Parameters

idx	index

Returns

bool true if weight is burrowed from outside

Definition at line 383 of file layer_context.cpp.

8.334.3.37 outputHasGradient()

```
bool nntrainer::RunLayerContext::outputHasGradient (
            unsigned int idx ) const
```

check if the weight has gradient

check if the output has gradient

Parameters

idx	Identifier of the weight

Returns

true if weight has gradient, else false

Parameters

idx	Identifier of the output

Returns

true if output has gradient, else false

Definition at line 251 of file layer_context.cpp.

8.334.3.38 readyToUse()

```
bool nntrainer::RunLayerContext::readyToUse ( ) const
```

check if run context is set and is ready to use

Returns

true if ready, else false

assumption:

1. there must be at least 1 input
2. the setter set everything at once

Definition at line 473 of file layer_context.cpp.

8.334.3.39 setBatch()

```
void nntrainer::RunLayerContext::setBatch (
            unsigned int batch )
```

Set the batch for the run context.

Parameters

batch	Update batch size

Definition at line 414 of file layer_context.cpp.

8.334.3.40 setLoss()

```
void nntrainer::RunLayerContext::setLoss (
            float val )  [inline]
```

update loss by the layer

Parameters

val	updated loss value

Note

loss value is only used for loss layers. For non-loss layers, setting this value will have no change on the behavior of the model.

Definition at line 654 of file layer_context.h.

8.334.3.41 tensorHasGradient()

```
bool nntrainer::RunLayerContext::tensorHasGradient (
            unsigned int idx ) const
```

check if the tensor has gradient

Parameters

idx	Identifier of the tensor

Returns

true if tensor has gradient, else false

Definition at line 379 of file layer_context.cpp.

8.334.3.42 updateTensor()

```
void nntrainer::RunLayerContext::updateTensor (
            unsigned int idx,
            unsigned int batch )
```

Update the dimensions for a requested tensor.

Parameters

idx	index of the tensor (identifier)
batch	Updated batch size

Definition at line 427 of file layer_context.cpp.

8.334.3.43 validate()

```
bool nntrainer::RunLayerContext::validate (
            bool skip_input = false,
            bool skip_label = false )
```

validates the run context after run

Parameters

skip_input	skip verifying the input
skip_label	skip verifying the label

Returns

true if ready, else false

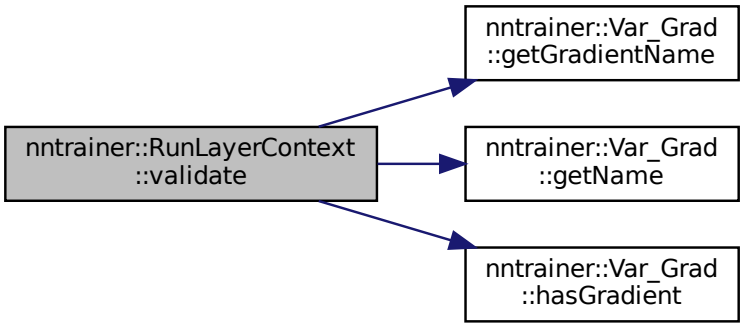
true if ready, else false

Note

a common mistake when using run_context is re-assigning the tensor references which leads to nasty bugs. This validation ensures that the tensors are not set mistakenly by verifying their unique names

Definition at line 489 of file layer_context.cpp.

Here is the call graph for this function:



8.334.3.44 weightHasGradient()

```
bool nntrainer::RunLayerContext::weightHasGradient (
            unsigned int idx ) const
```

check if the weight has gradient

Parameters

idx	Identifier of the weight

Returns

true if weight has gradient, else false

Definition at line 214 of file layer_context.cpp.

The documentation for this class was generated from the following files:

- layers/layer_context.h
- layers/layer_context.cpp

8.335 nntrainer::RunOptimizerContext Class Reference

Public Member Functions

- RunOptimizerContext (Weight *w=nullptr, size_t iter=0, double lr=0.0)
  Construct a new Run Optimizer Context object.
- Tensor & getWeight () const
  Get the Weight tensor object.
- Tensor & getGradient () const
  Get the Weight Gradient tensor object.
- Tensor & getOptimizerVariable (unsigned int idx) const
  Get the optimizer variable associated to this weight.
- bool readyToUse () const
  Check if run context is set and is ready to use.
- void applyGradient (double lr) const
  Apply the gradient with the given learning rate.
- size_t getIteration () const
  Get the current iteration value.
- double getLearningRate () const
  Get the current iteration value.

8.335.1 Detailed Description

Definition at line 31 of file optimizer_context.h.

8.335.2 Constructor & Destructor Documentation

8.335.2.1 RunOptimizerContext()

```
nntrainer::RunOptimizerContext::RunOptimizerContext (
            Weight * w = nullptr,
            size_t iter = 0,
            double lr = 0.0 )  [inline]
```

Construct a new Run Optimizer Context object.

Definition at line 37 of file optimizer_context.h.

8.335.3 Member Function Documentation

8.335.3.1 applyGradient()

```
void nntrainer::RunOptimizerContext::applyGradient (
            double lr ) const
```

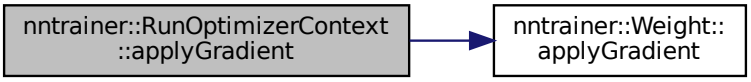
Apply the gradient with the given learning rate.

Parameters

lr	learning rate

Definition at line 42 of file optimizer_context.cpp.

Here is the call graph for this function:



8.335.3.2 getGradient()

```
Tensor & nntrainer::RunOptimizerContext::getGradient ( ) const
```

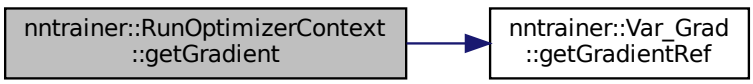
Get the Weight Gradient tensor object.

Returns

Tensor& Reference to the weight grad tensor

Definition at line 28 of file optimizer_context.cpp.

Here is the call graph for this function:



8.335.3.3 getIteration()

```
size_t nntrainer::RunOptimizerContext::getIteration ( ) const  [inline]
```

Get the current iteration value.

Returns

iteration value

Definition at line 83 of file optimizer_context.h.

8.335.3.4 getLearningRate()

```
double nntrainer::RunOptimizerContext::getLearningRate ( ) const  [inline]
```

Get the current iteration value.

Returns

iteration value

Definition at line 90 of file optimizer_context.h.

8.335.3.5 getOptimizerVariable()

```
Tensor & nntrainer::RunOptimizerContext::getOptimizerVariable (
            unsigned int idx ) const
```

Get the optimizer variable associated to this weight.

Parameters

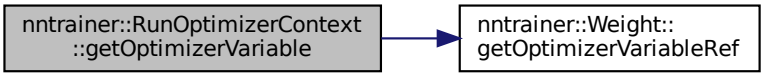
idx	Identifier of the associated weight

Returns

Tensor& Reference to the optimizer variable

Definition at line 35 of file optimizer_context.cpp.

Here is the call graph for this function:



8.335.3.6 getWeight()

```
Tensor & nntrainer::RunOptimizerContext::getWeight ( ) const
```

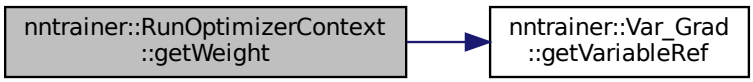
Get the Weight tensor object.

Returns

Tensor& Reference to the weight tensor

Definition at line 21 of file optimizer_context.cpp.

Here is the call graph for this function:



8.335.3.7 readyToUse()

```
bool nntrainer::RunOptimizerContext::readyToUse ( ) const  [inline]
```

Check if run context is set and is ready to use.

Returns

true if ready, else false

Definition at line 69 of file optimizer_context.h.

The documentation for this class was generated from the following files:

- optimizers/optimizer_context.h
- optimizers/optimizer_context.cpp

8.336 nntrainer::Sample Class Reference

Sample class which views the memory for a single sample.

`#include <data_iteration.h>`

Public Member Functions

- Sample (const Iteration &iter, unsigned int batch)
  Construct a new Sample object.
- **Sample** (const Sample &rhs)=delete
- Sample & **operator=** (const Sample &rhs)=delete
- **Sample** (Sample &&rhs)=default
- Sample & **operator=** (Sample &&rhs)=default
- std::vector< Tensor > & getInputsRef ()
  Get the Input Reference object.
- const std::vector< Tensor > & getInputsRef () const
  Get the Input Reference object.
- std::vector< Tensor > & getLabelsRef ()
  Get the Label Reference object.
- const std::vector< Tensor > & getLabelsRef () const
  Get the Label Reference object.

8.336.1 Detailed Description

Sample class which views the memory for a single sample.

Definition at line 137 of file data_iteration.h.

8.336.2 Constructor & Destructor Documentation

8.336.2.1 Sample()

```
nntrainer::Sample::Sample (
            const Iteration & iter,
            unsigned int batch )
```

Construct a new Sample object.

Note

the batch dimension will be ignored to make a single sample

Parameters

iter	iteration obejcts
batch	nth batch to create the sample

Definition at line 96 of file data_iteration.cpp.

8.336.3 Member Function Documentation

8.336.3.1 getInputsRef() [1/2]

```
std::vector<Tensor>& nntrainer::Sample::getInputsRef ( )  [inline]
```

Get the Input Reference object.

Returns

std::vector<Tensor>& input

Definition at line 159 of file data_iteration.h.

8.336.3.2 getInputsRef() [2/2]

```
const std::vector<Tensor>& nntrainer::Sample::getInputsRef ( ) const  [inline]
```

Get the Input Reference object.

Returns

const std::vector<Tensor>& input

Definition at line 166 of file data_iteration.h.

8.336.3.3 getLabelsRef() [1/2]

```
std::vector<Tensor>& nntrainer::Sample::getLabelsRef ( )  [inline]
```

Get the Label Reference object.

Returns

std::vector<Tensor>& label

Definition at line 173 of file data_iteration.h.

8.336.3.4 getLabelsRef() [2/2]

```
const std::vector<Tensor>& nntrainer::Sample::getLabelsRef ( ) const  [inline]
```

Get the Label Reference object.

Returns

const std::vector<Tensor>& label

Definition at line 180 of file data_iteration.h.

The documentation for this class was generated from the following files:

- dataset/data_iteration.h
- dataset/data_iteration.cpp

8.337 tflite::ScatterNdOptionsBuilder Struct Reference

Public Types

- typedef ScatterNdOptions **Table**

Public Member Functions

- **ScatterNdOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ScatterNdOptionsBuilder & **operator=** (const ScatterNdOptionsBuilder &)
- flatbuffers::Offset< ScatterNdOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.337.1 Detailed Description

Definition at line 6882 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.338 nntrainer::ScopedView< T > Class Template Reference

A view container that calls a callback on destruct.

```
#include <iteration_queue.h>
```

Public Member Functions

- ScopedView (T *data_, std::function< void(void)> &&on_notify_=nullptr, std::function< void(void)> &&on_error_=nullptr)

  Construct a new Scoped View object.
- **ScopedView** (const ScopedView &rhs)=delete
- ScopedView & **operator=** (const ScopedView &rhs)=delete
- **ScopedView** (ScopedView &&rhs)=default
- ScopedView & **operator=** (ScopedView &&rhs)=default
- bool isEmpty ()

  check if scoped view contains any underlying data
- ~ScopedView ()

  Destroy the Scoped View object, callback is called at this time.
- T & get ()

  get the underlying data
- const T & get () const

  get the underlying data

8.338.1 Detailed Description

template<typename T>
class nntrainer::ScopedView< T >

A view container that calls a callback on destruct.

Note

the callback must be noexcept, and the given underlying data must outlive the lifetime of this class

Template Parameters

T	underlying type

Definition at line 108 of file iteration_queue.h.

8.338.2 Constructor & Destructor Documentation

8.338.2.1 ScopedView()

```
template<typename T >
nntrainer::ScopedView< T >::ScopedView (
            T * data_,
            std::function< void(void)> && on_notify_ = nullptr,
            std::function< void(void)> && on_error_ = nullptr )  [inline]
```

Construct a new Scoped View object.

Parameters

*data_*	reference of the underlying data
*on_↩ notify_*	callback to be called on exit
*on_↩ error_*	callback to be called on error

Definition at line 117 of file iteration_queue.h.

8.338.2.2 ~ScopedView()

```
template<typename T >
nntrainer::ScopedView< T >::~ScopedView ( )  [inline]
```

Destroy the Scoped View object, callback is called at this time.

Definition at line 140 of file iteration_queue.h.

8.338.3 Member Function Documentation

8.338.3.1 get() [1/2]

```
template<typename T >
T& nntrainer::ScopedView< T >::get ( )  [inline]
```

get the underlying data

Returns

T & reference to the underlying data

Definition at line 161 of file iteration_queue.h.

8.338.3.2 get() [2/2]

```
template<typename T >
const T& nntrainer::ScopedView< T >::get ( ) const  [inline]
```

get the underlying data

Returns

T & reference to the underlying data

Definition at line 168 of file iteration_queue.h.

8.338.3.3 isEmpty()

```
template<typename T >
bool nntrainer::ScopedView< T >::isEmpty ( )  [inline]
```

check if scoped view contains any underlying data

Returns

bool if data is empty

Definition at line 134 of file iteration_queue.h.

The documentation for this class was generated from the following file:

- dataset/iteration_queue.h

8.339 tflite::SegmentSumOptionsBuilder Struct Reference

Public Types

- typedef SegmentSumOptions **Table**

Public Member Functions

- **SegmentSumOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SegmentSumOptionsBuilder & **operator=** (const SegmentSumOptionsBuilder &)
- flatbuffers::Offset< SegmentSumOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.339.1 Detailed Description

Definition at line 6972 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.340 tflite::SelectOptionsBuilder Struct Reference

Public Types

- typedef SelectOptions **Table**

Public Member Functions

- **SelectOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SelectOptionsBuilder & **operator=** (const SelectOptionsBuilder &)
- flatbuffers::Offset< SelectOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.340.1 Detailed Description

Definition at line 5396 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.341 tflite::SelectV2OptionsBuilder Struct Reference

Public Types

- typedef SelectV2Options **Table**

Public Member Functions

- **SelectV2OptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SelectV2OptionsBuilder & **operator=** (const SelectV2OptionsBuilder &)
- flatbuffers::Offset< SelectV2Options > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.341.1 Detailed Description

Definition at line 6912 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.342 tflite::SequenceRNNOptionsBuilder Struct Reference

Public Types

- typedef SequenceRNNOptions **Table**

Public Member Functions

- void **add_time_major** (bool time_major)
- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_asymmetric_quantize_inputs** (bool asymmetric_quantize_inputs)
- **SequenceRNNOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SequenceRNNOptionsBuilder & **operator=** (const SequenceRNNOptionsBuilder &)
- flatbuffers::Offset< SequenceRNNOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.342.1 Detailed Description

Definition at line 3221 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.343 tflite::ShapeOptionsBuilder Struct Reference

Public Types

- typedef ShapeOptions **Table**

Public Member Functions

- void **add_out_type** (tflite::TensorType out_type)
- **ShapeOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ShapeOptionsBuilder & **operator=** (const ShapeOptionsBuilder &)
- flatbuffers::Offset< ShapeOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.343.1 Detailed Description

Definition at line 5657 of file tf_schema_generated.h.

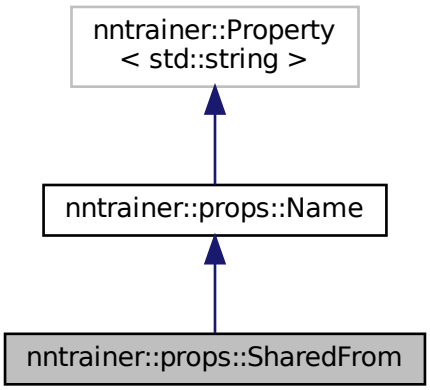
The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

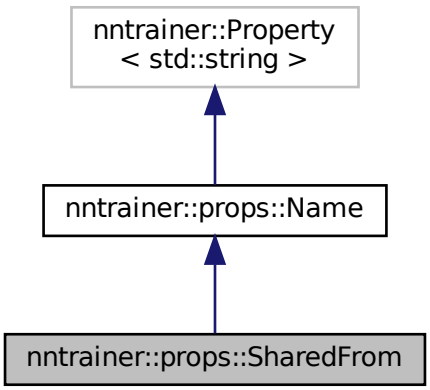
8.344 nntrainer::props::SharedFrom Class Reference

properties for shared from

Inheritance diagram for nntrainer::props::SharedFrom:



Collaboration diagram for nntrainer::props::SharedFrom:



Public Types

- using prop_tag = str_prop_tag

Static Public Attributes

- static constexpr const char * key = "shared_from"

Additional Inherited Members

8.344.1 Detailed Description

properties for shared from

Definition at line 113 of file layer_node.cpp.

8.344.2 Member Typedef Documentation

8.344.2.1 prop_tag

```
using nntrainer::props::SharedFrom::prop_tag = str_prop_tag
```

property type

Definition at line 116 of file layer_node.cpp.

8.344.3 Member Data Documentation

8.344.3.1 key

```
constexpr const char* nntrainer::props::SharedFrom::key = "shared_from"  [static], [constexpr]
```

unique key to access

Definition at line 115 of file layer_node.cpp.

The documentation for this class was generated from the following file:

- layers/layer_node.cpp

8.345 tflite::SignatureDefBuilder Struct Reference

Public Types

- typedef SignatureDef **Table**

Public Member Functions

- void **add_inputs** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::TensorMap >>> inputs)
- void **add_outputs** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::TensorMap >>> outputs)
- void **add_method_name** (flatbuffers::Offset< flatbuffers::String > method_name)
- void **add_key** (flatbuffers::Offset< flatbuffers::String > key)
- **SignatureDefBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SignatureDefBuilder & **operator=** (const SignatureDefBuilder &)
- flatbuffers::Offset< SignatureDef > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.345.1 Detailed Description

Definition at line 8385 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.346 tflite::SkipGramOptionsBuilder Struct Reference

Public Types

- typedef SkipGramOptions **Table**

Public Member Functions

- void **add_ngram_size** (int32_t ngram_size)
- void **add_max_skip_size** (int32_t max_skip_size)
- void **add_include_all_ngrams** (bool include_all_ngrams)
- **SkipGramOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SkipGramOptionsBuilder & **operator=** (const SkipGramOptionsBuilder &)
- flatbuffers::Offset< SkipGramOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.346.1 Detailed Description

Definition at line 4303 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.347 tflite::SliceOptionsBuilder Struct Reference

Public Types

- typedef SliceOptions **Table**

Public Member Functions

- **SliceOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SliceOptionsBuilder & **operator=** (const SliceOptionsBuilder &)
- flatbuffers::Offset< SliceOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.347.1 Detailed Description

Definition at line 5426 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

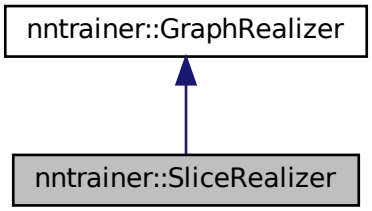
- compiler/tf_schema_generated.h

8.348 nntrainer::SliceRealizer Class Reference

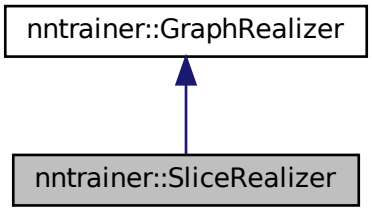
Graph realizer class which slice graph representation.

`#include <slice_realizer.h>`

Inheritance diagram for nntrainer::SliceRealizer:



Collaboration diagram for nntrainer::SliceRealizer:



Public Member Functions

- SliceRealizer (const std::vector< Connection > &start_connections, const std::vector< Connection > &end_connections)

  Construct a new Slice Realizer object.
- ~SliceRealizer ()

  Destroy the Graph Realizer object.
- GraphRepresentation realize (const GraphRepresentation &reference) override

  graph realizer creates a new graph based on the reference

8.348.1 Detailed Description

Graph realizer class which slice graph representation.

Definition at line 30 of file slice_realizer.h.

8.348.2 Constructor & Destructor Documentation

8.348.2.1 SliceRealizer()

```
nntrainer::SliceRealizer::SliceRealizer (
            const std::vector< Connection > & start_connections,
            const std::vector< Connection > & end_connections )
```

Construct a new Slice Realizer object.

Parameters

start_connections	start layers
end_connections	end layers

discard index information as it is not needed as it is not really needed

Definition at line 22 of file slice_realizer.cpp.

8.348.2.2 ~SliceRealizer()

```
nntrainer::SliceRealizer::~SliceRealizer ( )
```

Destroy the Graph Realizer object.

Definition at line 36 of file slice_realizer.cpp.

8.348.3 Member Function Documentation

8.348.3.1 realize()

```
GraphRepresentation nntrainer::SliceRealizer::realize (
            const GraphRepresentation & reference ) [override], [virtual]
```

graph realizer creates a new graph based on the reference

Note

for each layer in start_layers, start dfs, if traversal meets an end layers, node is added to an ordered set.

Exceptions

std::invalid_argument	if created GraphRepresentation is empty

< set this if not visited

< set this if visited

< set this if it is to be added

Note

mp has to be ordered map to keep the ordering of the nodes in the graph

map point

setup children before filling in the end layers

if the give node is the end node in the graph

add node to be included to subgraph

dfs function to perform depth-first search recursively with tracking

if node already added or end node, add the current stack to be added to the subgraph

if node is visited, return

run dfs on all the children

run dfs from all the starting layers

created the subgraph

Note

: iterate over reference than over mp to ensure the correct ordering of layers

Implements nntrainer::GraphRealizer.

Definition at line 39 of file slice_realizer.cpp.

The documentation for this class was generated from the following files:

- compiler/slice_realizer.h
- compiler/slice_realizer.cpp

8.349 tflite::SoftmaxOptionsBuilder Struct Reference

Public Types

- typedef SoftmaxOptions **Table**

Public Member Functions

- void **add_beta** (float beta)
- **SoftmaxOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SoftmaxOptionsBuilder & **operator=** (const SoftmaxOptionsBuilder &)
- flatbuffers::Offset< SoftmaxOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.349.1 Detailed Description

Definition at line 3417 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.350 tflite::SpaceToBatchNDOptionsBuilder Struct Reference

Public Types

- typedef SpaceToBatchNDOptions **Table**

Public Member Functions

- **SpaceToBatchNDOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SpaceToBatchNDOptionsBuilder & **operator=** (const SpaceToBatchNDOptionsBuilder &)
- flatbuffers::Offset< SpaceToBatchNDOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.350.1 Detailed Description

Definition at line 4226 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.351 tflite::SpaceToDepthOptionsBuilder Struct Reference

Public Types

- typedef SpaceToDepthOptions **Table**

Public Member Functions

- void **add_block_size** (int32_t block_size)
- **SpaceToDepthOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SpaceToDepthOptionsBuilder & **operator=** (const SpaceToDepthOptionsBuilder &)
- flatbuffers::Offset< SpaceToDepthOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.351.1 Detailed Description

Definition at line 4355 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.352 tflite::SparseIndexVectorTraits< T > Struct Template Reference

Static Public Attributes

- static const SparseIndexVector **enum_value** = SparseIndexVector_NONE

8.352.1 Detailed Description

template<typename T>
struct tflite::SparseIndexVectorTraits< T >

Definition at line 532 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.353 tflite::SparseIndexVectorTraits< tflite::Int32Vector > Struct Reference

Static Public Attributes

- static const SparseIndexVector **enum_value** = SparseIndexVector_Int32Vector

8.353.1 Detailed Description

Definition at line 536 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.354 tflite::SparseIndexVectorTraits< tflite::Uint16Vector > Struct Reference

Static Public Attributes

- static const SparseIndexVector **enum_value** = SparseIndexVector_Uint16Vector

8.354.1 Detailed Description

Definition at line 540 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.355 tflite::SparseIndexVectorTraits< tflite::Uint8Vector > Struct Reference

Static Public Attributes

- static const SparseIndexVector **enum_value** = SparseIndexVector_Uint8Vector

8.355.1 Detailed Description

Definition at line 544 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.356 tflite::SparseToDenseOptionsBuilder Struct Reference

Public Types

- typedef SparseToDenseOptions **Table**

Public Member Functions

- void **add_validate_indices** (bool validate_indices)
- **SparseToDenseOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SparseToDenseOptionsBuilder & **operator=** (const SparseToDenseOptionsBuilder &)
- flatbuffers::Offset< SparseToDenseOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.356.1 Detailed Description

Definition at line 5555 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.357 tflite::SparsityParametersBuilder Struct Reference

Public Types

- typedef SparsityParameters **Table**

Public Member Functions

- void **add_traversal_order** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> traversal_order)
- void **add_block_map** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> block_map)
- void **add_dim_metadata** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::Dimension↩ Metadata >>> dim_metadata)
- **SparsityParametersBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SparsityParametersBuilder & **operator=** (const SparsityParametersBuilder &)
- flatbuffers::Offset< SparsityParameters > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.357.1 Detailed Description

Definition at line 2482 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

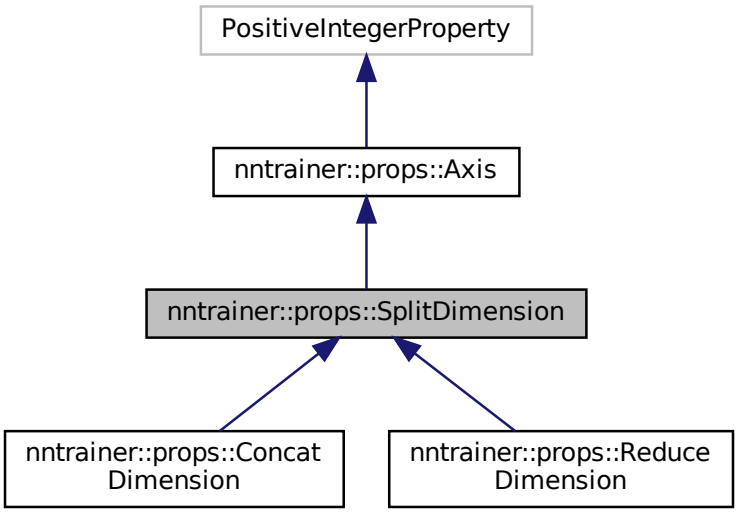
- compiler/tf_schema_generated.h

8.358 nntrainer::props::SplitDimension Class Reference

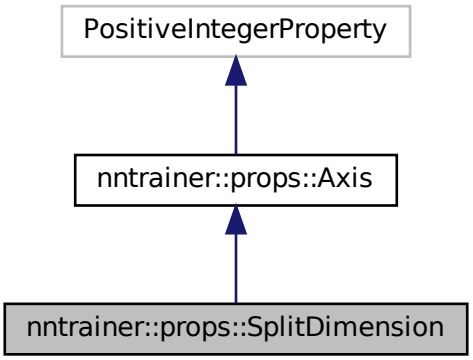
SplitDimension property, dimension along which to split the input.

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::SplitDimension:



Collaboration diagram for nntrainer::props::SplitDimension:



Public Member Functions

- bool isValid (const unsigned int &value) const override
  check if given value is valid

Additional Inherited Members

8.358.1 Detailed Description

SplitDimension property, dimension along which to split the input.

Definition at line 290 of file common_properties.h.

8.358.2 Member Function Documentation

8.358.2.1 isValid()

```
bool nntrainer::props::SplitDimension::isValid (
            const unsigned int & value ) const  [override]
```

check if given value is valid

Parameters

v	value to check

Return values

true	if it is greater than 0 and smaller than ml::train::TensorDim::MAXDIM
false	if it is samller or equal to 0 or greate than ml::train::TensorDim::MAXDIM

Definition at line 103 of file common_properties.cpp.

The documentation for this class was generated from the following files:

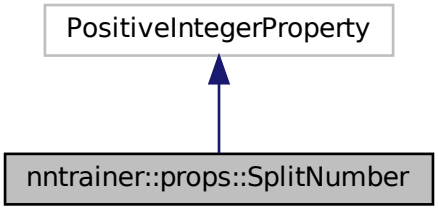
- layers/common_properties.h
- layers/common_properties.cpp

8.359 nntrainer::props::SplitNumber Class Reference

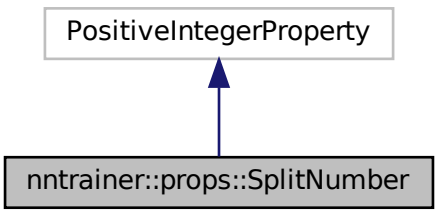
split number property, split number indicates how many numbers of outs are generated by spliting the input dimension

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::SplitNumber:



Collaboration diagram for nntrainer::props::SplitNumber:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key

8.359.1 Detailed Description

split number property, split number indicates how many numbers of outs are generated by spliting the input dimension

Definition at line 258 of file common_properties.h.

8.359.2 Member Typedef Documentation

8.359.2.1 prop_tag

```
using nntrainer::props::SplitNumber::prop_tag = uint_prop_tag
```

property type

Definition at line 262 of file common_properties.h.

8.359.3 Member Data Documentation

8.359.3.1 key

```
constexpr const char* nntrainer::props::SplitNumber::key  [static], [constexpr]
```

Initial value:

```
=
    "split_number"
```

unique key to access

Definition at line 260 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.360 tflite::SplitOptionsBuilder Struct Reference

Public Types

- typedef SplitOptions **Table**

Public Member Functions

- void **add_num_splits** (int32_t num_splits)
- **SplitOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SplitOptionsBuilder & **operator=** (const SplitOptionsBuilder &)
- flatbuffers::Offset< SplitOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.360.1 Detailed Description

Definition at line 4831 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.361 tflite::SplitVOptionsBuilder Struct Reference

Public Types

- typedef SplitVOptions **Table**

Public Member Functions

- void **add_num_splits** (int32_t num_splits)
- **SplitVOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SplitVOptionsBuilder & **operator=** (const SplitVOptionsBuilder &)
- flatbuffers::Offset< SplitVOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.361.1 Detailed Description

Definition at line 4873 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.362 tflite::SquaredDifferenceOptionsBuilder Struct Reference

Public Types

- typedef SquaredDifferenceOptions **Table**

Public Member Functions

- **SquaredDifferenceOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SquaredDifferenceOptionsBuilder & **operator=** (const SquaredDifferenceOptionsBuilder &)
- flatbuffers::Offset< SquaredDifferenceOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.362.1 Detailed Description

Definition at line 6342 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.363 tflite::SquareOptionsBuilder Struct Reference

Public Types

- typedef SquareOptions **Table**

Public Member Functions

- **SquareOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SquareOptionsBuilder & **operator=** (const SquareOptionsBuilder &)
- flatbuffers::Offset< SquareOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.363.1 Detailed Description

Definition at line 6150 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.364 tflite::SqueezeOptionsBuilder Struct Reference

Public Types

- typedef SqueezeOptions **Table**

Public Member Functions

- void **add_squeeze_dims** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> squeeze_dims)
- **SqueezeOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SqueezeOptionsBuilder & **operator=** (const SqueezeOptionsBuilder &)
- flatbuffers::Offset< SqueezeOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.364.1 Detailed Description

Definition at line 4780 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.365 nntrainer::SrcSharedTensor Class Reference

Source of the shared tensor.

Public Member Functions

- SrcSharedTensor ()

  Constructor for the class.
- **SrcSharedTensor** (const Tensor *tensor, size_t offset)
- const Tensor * tensor () const

  Get the allocated src tensor.
- size_t offset () const

  Get the offset from the source tensor.

8.365.1 Detailed Description

Source of the shared tensor.

Definition at line 119 of file tensor.cpp.

The documentation for this class was generated from the following file:

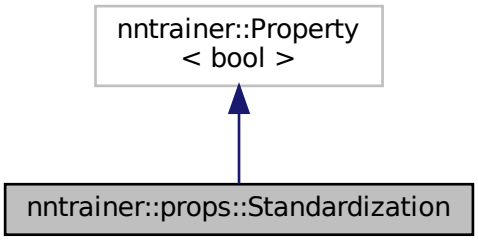
- tensor/tensor.cpp

8.366 nntrainer::props::Standardization Class Reference

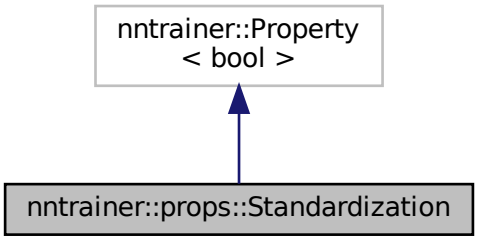
Standardization property, standardization standardize the input to be mean 0 and std 1 if true.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Standardization:



Collaboration diagram for nntrainer::props::Standardization:



Public Types

- using **prop_tag** = bool_prop_tag

Public Member Functions

- Standardization (bool value=false)

  Construct a new Standardization object.

Static Public Attributes

- static constexpr const char * **key** = "standardization"

8.366.1 Detailed Description

Standardization property, standardization standardize the input to be mean 0 and std 1 if true.

Definition at line 162 of file common_properties.h.

8.366.2 Constructor & Destructor Documentation

8.366.2.1 Standardization()

```
nntrainer::props::Standardization::Standardization (
            bool value = false )
```

Construct a new Standardization object.

Definition at line 51 of file common_properties.cpp.

The documentation for this class was generated from the following files:

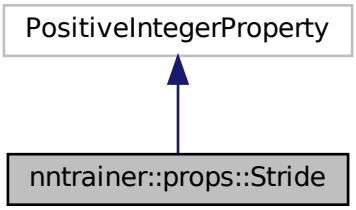
- layers/common_properties.h
- layers/common_properties.cpp

8.367 nntrainer::props::Stride Class Reference

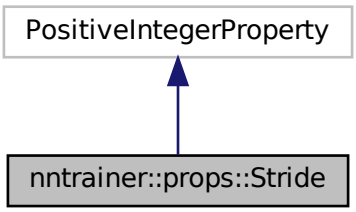
Stride property, stride is used to measure how much it will be slide the filter.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Stride:



Collaboration diagram for nntrainer::props::Stride:



Public Types

- using prop_tag = uint_prop_tag

Public Member Functions

- Stride (unsigned int value=1)

  Construct a new Stride object with a default value 1.

Static Public Attributes

- static constexpr const char * key = "stride"

8.367.1 Detailed Description

Stride property, stride is used to measure how much it will be slide the filter.

Definition at line 363 of file common_properties.h.

8.367.2 Member Typedef Documentation

8.367.2.1 prop_tag

```
using nntrainer::props::Stride::prop_tag = uint_prop_tag
```

property type

Definition at line 371 of file common_properties.h.

8.367.3 Constructor & Destructor Documentation

8.367.3.1 Stride()

```
nntrainer::props::Stride::Stride (
            unsigned int value = 1 )
```

Construct a new Stride object with a default value 1.

Definition at line 109 of file common_properties.cpp.

8.367.4 Member Data Documentation

8.367.4.1 key

```
constexpr const char* nntrainer::props::Stride::key = "stride"  [static], [constexpr]
```

unique key to access

Definition at line 370 of file common_properties.h.

The documentation for this class was generated from the following files:

- layers/common_properties.h
- layers/common_properties.cpp

8.368 tflite::StridedSliceOptionsBuilder Struct Reference

Public Types

- typedef StridedSliceOptions **Table**

Public Member Functions

- void **add_begin_mask** (int32_t begin_mask)
- void **add_end_mask** (int32_t end_mask)
- void **add_ellipsis_mask** (int32_t ellipsis_mask)
- void **add_new_axis_mask** (int32_t new_axis_mask)
- void **add_shrink_axis_mask** (int32_t shrink_axis_mask)
- **StridedSliceOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- StridedSliceOptionsBuilder & **operator=** (const StridedSliceOptionsBuilder &)
- flatbuffers::Offset< StridedSliceOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.368.1 Detailed Description

Definition at line 4935 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.369 tflite::SubGraphBuilder Struct Reference

Public Types

- typedef SubGraph **Table**

Public Member Functions

- void **add_tensors** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::Tensor >>> tensors)
- void **add_inputs** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> inputs)
- void **add_outputs** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> outputs)
- void **add_operators** (flatbuffers::Offset< flatbuffers::Vector< flatbuffers::Offset< tflite::Operator >>> operators)
- void **add_name** (flatbuffers::Offset< flatbuffers::String > name)
- **SubGraphBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SubGraphBuilder & **operator=** (const SubGraphBuilder &)
- flatbuffers::Offset< SubGraph > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.369.1 Detailed Description

Definition at line 8100 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.370 tflite::SubOptionsBuilder Struct Reference

Public Types

- typedef SubOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_pot_scale_int16** (bool pot_scale_int16)
- **SubOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SubOptionsBuilder & **operator=** (const SubOptionsBuilder &)
- flatbuffers::Offset< SubOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.370.1 Detailed Description

Definition at line 4444 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.371 tflite::SVDFOptionsBuilder Struct Reference

Public Types

- typedef SVDFOptions **Table**

Public Member Functions

- void **add_rank** (int32_t rank)
- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_asymmetric_quantize_inputs** (bool asymmetric_quantize_inputs)
- **SVDFOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- SVDFOptionsBuilder & **operator=** (const SVDFOptionsBuilder &)
- flatbuffers::Offset< SVDFOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.371.1 Detailed Description

Definition at line 3107 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.372 nntrainer::SwapDevice Class Reference

A device used to storing data with long access time.

`#include <swap_device.h>`

Public Member Functions

- SwapDevice (const std::string &name)

  SwapDevice default constructor.
- SwapDevice (const std::string &path, const std::string &name)

  SwapDevice default constructor.
- virtual ~SwapDevice ()=default

  SwapDevice destructor.
- void start (size_t size)

  Start device.
- void * getBuffer (off_t offset, size_t size, bool alloc_only=false)

  Allocate and get data.
- void putBuffer (void *ptr, bool dealloc_only=false)

  Deallocate and put data.
- void finish ()

  Close device.
- bool isOperating () const

  Check device is operating.
- const std::string getDevicePath () const

  Get device path.

Public Attributes

- const std::string swap_device_default_path = "."
  swap device default path

8.372.1 Detailed Description

A device used to storing data with long access time.

Definition at line 37 of file swap_device.h.

8.372.2 Constructor & Destructor Documentation

8.372.2.1 SwapDevice() [1/2]

```
nntrainer::SwapDevice::SwapDevice (
            const std::string & name )  [inline], [explicit]
```

SwapDevice default constructor.

Definition at line 49 of file swap_device.h.

8.372.2.2 SwapDevice() [2/2]

```
nntrainer::SwapDevice::SwapDevice (
            const std::string & path,
            const std::string & name )  [inline], [explicit]
```

SwapDevice default constructor.

Definition at line 57 of file swap_device.h.

8.372.2.3 ~SwapDevice()

```
virtual nntrainer::SwapDevice::~SwapDevice ( )  [virtual], [default]
```

SwapDevice destructor.

8.372.3 Member Function Documentation

8.372.3.1 finish()

```
void nntrainer::SwapDevice::finish ( )
```

Close device.

Definition at line 157 of file swap_device.cpp.

8.372.3.2 getBuffer()

```
void * nntrainer::SwapDevice::getBuffer (
            off_t offset,
            size_t size,
            bool alloc_only = false )
```

Allocate and get data.

Parameters

offset	Requested offset of swap device file
size	Requested size.
alloc_only	only allocate buffer without reading data

Returns

The pointer of the swap space

Definition at line 54 of file swap_device.cpp.

8.372.3.3 getDevicePath()

```
const std::string nntrainer::SwapDevice::getDevicePath ( ) const  [inline]
```

Get device path.

Returns

Device path

Definition at line 115 of file swap_device.h.

8.372.3.4 isOperating()

```
bool nntrainer::SwapDevice::isOperating ( ) const  [inline]
```

Check device is operating.

Returns

Device operating status

Definition at line 107 of file swap_device.h.

8.372.3.5 putBuffer()

```
void nntrainer::SwapDevice::putBuffer (
            void * ptr,
            bool dealloc_only = false )
```

Deallocate and put data.

Parameters

ptr	The pointer obtained from getBuffer
dealloc_only	only deallocate buffer without writing data

Definition at line 101 of file swap_device.cpp.

8.372.3.6 start()

```
void nntrainer::SwapDevice::start (
            size_t size )
```

Start device.

Parameters

size	The size of requested swap device space

Definition at line 28 of file swap_device.cpp.

8.372.4 Member Data Documentation

8.372.4.1 swap_device_default_path

```
const std::string nntrainer::SwapDevice::swap_device_default_path = "."
```

swap device default path

Definition at line 43 of file swap_device.h.

The documentation for this class was generated from the following files:

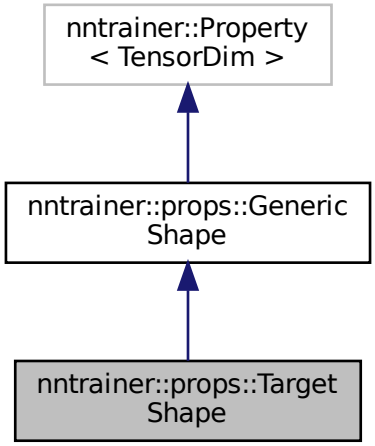
- tensor/swap_device.h
- tensor/swap_device.cpp

8.373 nntrainer::props::TargetShape Class Reference

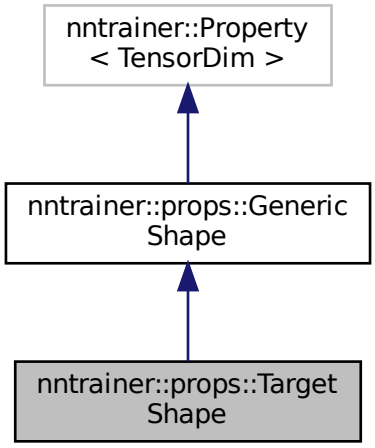
target shape property which saves a single tensor shape (practically, std::array<TargetShape> is used)

```
#include <common_properties.h>
```

Inheritance diagram for nntrainer::props::TargetShape:



Collaboration diagram for nntrainer::props::TargetShape:



Public Types

- using prop_tag = dimension_prop_tag

Static Public Attributes

- static constexpr const char * key

Additional Inherited Members

8.373.1 Detailed Description

target shape property which saves a single tensor shape (practically, std::array<TargetShape> is used)

Definition at line 1151 of file common_properties.h.

8.373.2 Member Typedef Documentation

8.373.2.1 prop_tag

```
using nntrainer::props::TargetShape::prop_tag = dimension_prop_tag
```

property type

Definition at line 1156 of file common_properties.h.

8.373.3 Member Data Documentation

8.373.3.1 key

```
constexpr const char* nntrainer::props::TargetShape::key  [static], [constexpr]
```

Initial value:

```
=
    "target_shape"
```

unique key to access

Definition at line 1154 of file common_properties.h.

The documentation for this class was generated from the following file:

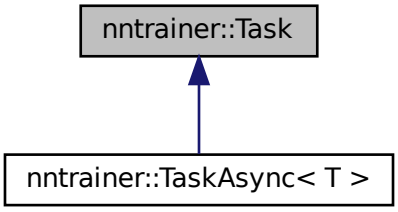
- layers/common_properties.h

8.374 nntrainer::Task Class Reference

task class

```
#include <task.h>
```

Inheritance diagram for nntrainer::Task:



Public Types

- enum Type { **SYNC**, **ASYNC** }
  task type
- enum State { **CREATED**, **STARTED**, **PROCESSING**, **DONE** }
  task state
- using Work = std::function< int(std::atomic_bool &running, void *data)>
  work function

Public Member Functions

- Task (Work w, void ∗user_data)
  Task constructor.
- virtual ∼Task ()=default
  Task destructor.
- virtual const Task::Type getType (void)
  Get type of task.
- bool started (void)
  Task destructor.
- bool done (void)
  Task destructor.
- const Work getWork (void)
  Get work of task.
- void ∗ getData (void)
  Task destructor.
- void setState (State s)
  Set task state.

8.374.1 Detailed Description

task class

Definition at line 27 of file task.h.

8.374.2 Constructor & Destructor Documentation

8.374.2.1 Task()

```
nntrainer::Task::Task (
            Work w,
            void * user_data )  [inline], [explicit]
```

Task constructor.

Definition at line 56 of file task.h.

8.374.2.2 ∼Task()

```
virtual nntrainer::Task::~Task ( )  [virtual], [default]
```

Task destructor.

8.374.3 Member Function Documentation

8.374.3.1 done()

```
bool nntrainer::Task::done (
            void  )  [inline]
```

Task destructor.

Returns

true if state is after done, else false

Definition at line 86 of file task.h.

8.374.3.2 getData()

```
void* nntrainer::Task::getData (
            void  )  [inline]
```

Task destructor.

Returns

user data

Definition at line 100 of file task.h.

8.374.3.3 getType()

```
virtual const Task::Type nntrainer::Task::getType (
            void  )  [inline], [virtual]
```

Get type of task.

Returns

type of task Task::Type

Reimplemented in nntrainer::TaskAsync< T >.

Definition at line 72 of file task.h.

8.374.3.4 getWork()

```
const Work nntrainer::Task::getWork (
            void  )  [inline]
```

Get work of task.

Returns

work function Task::Work

Definition at line 93 of file task.h.

8.374.3.5 setState()

```
void nntrainer::Task::setState (
            State s )  [inline]
```

Set task state.

Parameters

s	state to be set

Definition at line 107 of file task.h.

8.374.3.6 started()

```
bool nntrainer::Task::started (
            void  )  [inline]
```

Task destructor.

Returns

true if state is after started, else false

Definition at line 79 of file task.h.

The documentation for this class was generated from the following file:

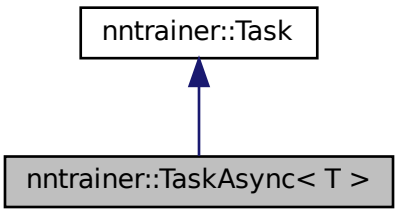
- tensor/task.h

8.375 nntrainer::TaskAsync< T > Class Template Reference

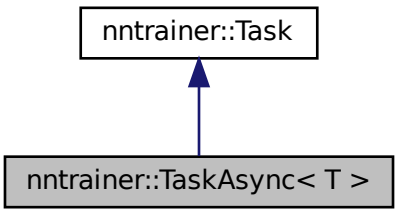
Async task class.

`#include <task.h>`

Inheritance diagram for nntrainer::TaskAsync< T >:



Collaboration diagram for nntrainer::TaskAsync< T >:



Public Types

- enum **Priority** { **HIGH**, **MID**, **LOW** }

Public Member Functions

- TaskAsync (Work work, void *user_data)

  TaskAsync constructor.
- virtual ~TaskAsync ()=default

  TaskAsync destructor.
- virtual const Task::Type getType (void) override

  Get type of task.
- virtual void setTimeout (int64_t time)

Set timeout.

- virtual int64_t getTimeout (void) const

  Get timeout.

- virtual void setPriority (Priority pri)

  Set priority.

- virtual Priority getPriority ()

  Get priority.

8.375.1 Detailed Description

template<typename T = std::chrono::milliseconds>
class nntrainer::TaskAsync< T >

Async task class.

Definition at line 120 of file task.h.

8.375.2 Constructor & Destructor Documentation

8.375.2.1 TaskAsync()

```
template<typename T = std::chrono::milliseconds>
nntrainer::TaskAsync< T >::TaskAsync (
            Work work,
            void * user_data )  [inline], [explicit]
```

TaskAsync constructor.

Definition at line 132 of file task.h.

8.375.2.2 ~TaskAsync()

```
template<typename T = std::chrono::milliseconds>
virtual nntrainer::TaskAsync< T >::~TaskAsync ( )  [virtual], [default]
```

TaskAsync destructor.

8.375.3 Member Function Documentation

8.375.3.1 getPriority()

```
template<typename T = std::chrono::milliseconds>
virtual Priority nntrainer::TaskAsync< T >::getPriority ( )  [inline], [virtual]
```

Get priority.

Returns

priority of task

Definition at line 176 of file task.h.

8.375.3.2 getTimeout()

```
template<typename T = std::chrono::milliseconds>
virtual int64_t nntrainer::TaskAsync< T >::getTimeout (
            void  ) const  [inline], [virtual]
```

Get timeout.

Parameters

time	timeout in T

Definition at line 162 of file task.h.

8.375.3.3 getType()

```
template<typename T = std::chrono::milliseconds>
virtual const Task::Type nntrainer::TaskAsync< T >::getType (
            void  )  [inline], [override], [virtual]
```

Get type of task.

Returns

type of task Task::Type

Reimplemented from nntrainer::Task.

Definition at line 148 of file task.h.

8.375.3.4 setPriority()

```
template<typename T = std::chrono::milliseconds>
virtual void nntrainer::TaskAsync< T >::setPriority (
            Priority pri )  [inline], [virtual]
```

Set priority.

Parameters

pri	priority to be set

Definition at line 169 of file task.h.

8.375.3.5 setTimeout()

```
template<typename T = std::chrono::milliseconds>
virtual void nntrainer::TaskAsync< T >::setTimeout (
             int64_t time )  [inline], [virtual]
```

Set timeout.

Parameters

time	timeout in T

Definition at line 155 of file task.h.

The documentation for this class was generated from the following file:

- tensor/task.h

8.376 nntrainer::TaskExecutor Class Reference

task executor class

```
#include <task_executor.h>
```

Public Types

- enum CompleteStatus { **SUCCESS**, **FAIL_CANCEL**, **FAIL_TIMEOUT**, **FAIL** }

  Complete error types.
- using CompleteCallback = std::function< void(int, CompleteStatus)>
- template<typename T = std::chrono::milliseconds>
  using TaskInfo = std::tuple< int, std::shared_ptr< TaskAsync< T > >, CompleteCallback, std::atomic_bool, std::promise< CompleteStatus > >

Public Member Functions

- TaskExecutor (const std::string &name="")

  TaskExecutor constructor.
- virtual ~TaskExecutor ()

  TaskExecutor destructor.
- virtual int run (std::shared_ptr< Task > task)

  Run task.
- template<typename T >
  int run (std::shared_ptr< TaskAsync< T >> task, CompleteCallback callback)

  Run task asynchronously.
- virtual void cancel (int id)

  Cancel task.
- virtual void cancelAll (void)

  Cancel all task.
- virtual void clean (void)

  Clean all non-running tasks from managed list.

Protected Member Functions

- template<typename T >
  CompleteStatus worker (TaskInfo< T > &info)

  task worker for asynchronous task
- virtual CompleteStatus handleWork (std::atomic_bool &running, Task::Work &work, void *data)

  handle work for asynchronous task

Protected Attributes

- std::string **name**
- bool **run_thread**
- bool **wait_complete**
- std::list< TaskInfo<> > **task_queue**
- std::condition_variable **task_cv**
- std::condition_variable **comp_cv**
- std::thread **task_thread**
- std::mutex **task_mutex**

Static Protected Attributes

- static std::atomic_int32_t **ids**

8.376.1 Detailed Description

task executor class

Definition at line 35 of file task_executor.h.

8.376.2 Member Typedef Documentation

8.376.2.1 CompleteCallback

```
using nntrainer::TaskExecutor::CompleteCallback = std::function<void(int, CompleteStatus)>
```

(task id, status)

Definition at line 51 of file task_executor.h.

8.376.2.2 TaskInfo

```
template<typename T = std::chrono::milliseconds>
using nntrainer::TaskExecutor::TaskInfo = std::tuple<int, std::shared_ptr<TaskAsync<T> >,
CompleteCallback, std::atomic_bool, std::promise<CompleteStatus> >
```

(task id, task, complete callback, running, complete promise)

Definition at line 57 of file task_executor.h.

8.376.3 Constructor & Destructor Documentation

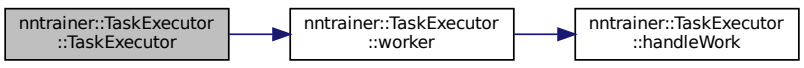
8.376.3.1 TaskExecutor()

```
nntrainer::TaskExecutor::TaskExecutor (
            const std::string & name = "" )  [explicit]
```

TaskExecutor constructor.

Definition at line 31 of file task_executor.cpp.

Here is the call graph for this function:



8.376.3.2 ~TaskExecutor()

```
nntrainer::TaskExecutor::~TaskExecutor ( )  [virtual]
```

TaskExecutor destructor.

Definition at line 60 of file task_executor.cpp.

8.376.4 Member Function Documentation

8.376.4.1 cancel()

```
void nntrainer::TaskExecutor::cancel (
            int id )  [virtual]
```

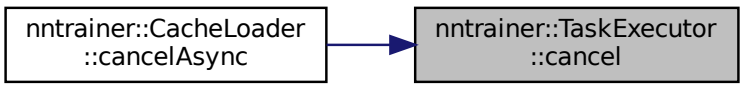
Cancel task.

Parameters

id	task id returned from TaskExecutorAsync::run

Definition at line 72 of file task_executor.cpp.

Here is the caller graph for this function:



8.376.4.2 clean()

```
void nntrainer::TaskExecutor::clean (
            void  )  [virtual]
```

Clean all non-running tasks from managed list.

Note

Do not use this inside of the complete or worker callback

Definition at line 88 of file task_executor.cpp.

8.376.4.3 handleWork()

```
TaskExecutor::CompleteStatus nntrainer::TaskExecutor::handleWork (
            std::atomic_bool & running,
            Task::Work & work,
            void * data )  [protected], [virtual]
```

handle work for asynchronous task

Parameters

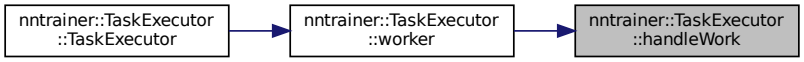
running	running flag
work	work function
data	data to be passed to work function

Returns

CompleteStatus

Definition at line 98 of file task_executor.cpp.

Here is the caller graph for this function:



8.376.4.4 run() [1/2]

```
int nntrainer::TaskExecutor::run (
            std::shared_ptr< Task > task )  [virtual]
```

Run task.

Parameters

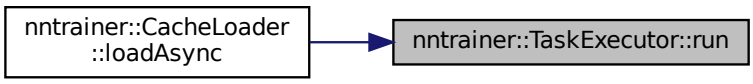
task	task to be run

Returns

0 on Success, else negative error

Definition at line 66 of file task_executor.cpp.

Here is the caller graph for this function:



8.376.4.5 run() [2/2]

```
template<typename T >
int nntrainer::TaskExecutor::run (
            std::shared_ptr< TaskAsync< T >> task,
            CompleteCallback callback )  [inline]
```

Run task asynchronously.

Parameters

task	async task to be run
callback	complete callback

Returns

id of requested task. negative on fail

Definition at line 88 of file task_executor.h.

8.376.4.6 worker()

```
template<typename T >
CompleteStatus nntrainer::TaskExecutor::worker (
            TaskInfo< T > & info )  [inline], [protected]
```

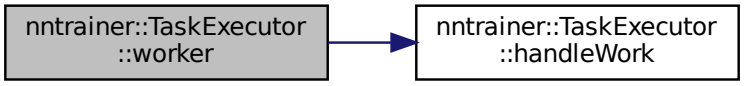
task worker for asynchronous task

Parameters

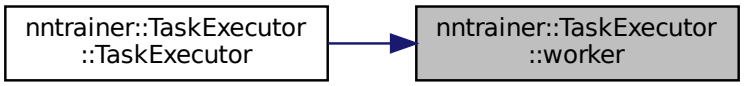
id	task id
task	asynchronous task

Definition at line 127 of file task_executor.h.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- tensor/task_executor.h
- tensor/task_executor.cpp

8.377 tflite::TensorBuilder Struct Reference

Public Types

- typedef Tensor **Table**

Public Member Functions

- void **add_shape** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> shape)
- void **add_type** (tflite::TensorType type)
- void **add_buffer** (uint32_t buffer)
- void **add_name** (flatbuffers::Offset< flatbuffers::String > name)
- void **add_quantization** (flatbuffers::Offset< tflite::QuantizationParameters > quantization)
- void **add_is_variable** (bool is_variable)
- void **add_sparsity** (flatbuffers::Offset< tflite::SparsityParameters > sparsity)
- void **add_shape_signature** (flatbuffers::Offset< flatbuffers::Vector< int32_t >> shape_signature)
- **TensorBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- TensorBuilder & **operator=** (const TensorBuilder &)
- flatbuffers::Offset< Tensor > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.377.1 Detailed Description

Definition at line 2589 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.378 tflite::TensorMapBuilder Struct Reference

Public Types

- typedef TensorMap **Table**

Public Member Functions

- void **add_name** (flatbuffers::Offset< flatbuffers::String > name)
- void **add_tensor_index** (uint32_t tensor_index)
- **TensorMapBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- TensorMapBuilder & **operator=** (const TensorMapBuilder &)
- flatbuffers::Offset< TensorMap > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.378.1 Detailed Description

Definition at line 8306 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.379 nntrainer::TensorSpecV2 Struct Reference

Tensor Specification which describes how this tensor should be allocated and managed.

`#include <tensor_wrap_specs.h>`

Public Types

- enum RequestType {
  RequestType::PLACEHOLDER, RequestType::UNIQUE, RequestType::READ_ONLY_VIEW, RequestType::MAYBE_MODIFYI
  RequestType::SHARED }

  Tensor is being managed by nntrainer, this enum defines how the value should be recognized inside nntrainer tensor managing scheme.

Public Attributes

- RequestType request_type = RequestType::UNIQUE
- std::string name
- TensorDim dim
- TensorLifespan ls
- Tensor::Initializer initializer
- unsigned int offset = 0u
- std::string reference_name
- std::vector< unsigned > additional_exec_order = {}

8.379.1 Detailed Description

Tensor Specification which describes how this tensor should be allocated and managed.

Definition at line 91 of file tensor_wrap_specs.h.

8.379.2 Member Enumeration Documentation

8.379.2.1 RequestType

```
enum nntrainer::TensorSpecV2::RequestType  [strong]
```

Tensor is being managed by nntrainer, this enum defines how the value should be recognized inside nntrainer tensor managing scheme.

Enumerator

PLACEHOLDER	Placeholder defines that nntrainer should never care about the memory inside the particualar tensor
UNIQUE	Unique means a simple tensor that will be owned explicitly the current request
READ_ONLY_VIEW	Readonly view defines a view of which ownership of *underlying* memory is at another tensor, also hinting nntrainer that the operation upon this particular tensor will never change value of the underlying memory
MAYBE_MODIFYING_VIEW	Maybe modifying view defines a (possible) view of which ownership of *underlying* memory is at another tensor, while hinting the nntrainer this tensor will do some modification of the underlying memory. nntrainer will try to make this particular tensor a view of the stated reference. If making a view of reference is likely to break the data integrity, nntrainer will request an independent memory slot, in this case, it is user's responsibility to copy the data.
SHARED	Shared defines a shared tensor ownership for the given identifier, it is user's responsibility to guarantee that dimension and initializer of shared tensor if exactly same as the user will be agnostic about when and who will actually request the certain tensor.

Definition at line 98 of file tensor_wrap_specs.h.

8.379.3 Member Data Documentation

8.379.3.1 additional_exec_order

```
std::vector<unsigned> nntrainer::TensorSpecV2::additional_exec_order = {}
```

ONLY FOR THE GRANULAR CONTROL OF LIFE OUTSIDE OF LAYER NODE

Todo make this as an opaque information with PIMPL

Definition at line 136 of file tensor_wrap_specs.h.

8.379.3.2 dim

```
TensorDim nntrainer::TensorSpecV2::dim
```

dimension

Definition at line 125 of file tensor_wrap_specs.h.

8.379.3.3 initializer

```
Tensor::Initializer nntrainer::TensorSpecV2::initializer
```

Initial value:

```
=
    Tensor::Initializer::NONE
```

initializer

Definition at line 127 of file tensor_wrap_specs.h.

8.379.3.4 ls

```
TensorLifespan nntrainer::TensorSpecV2::ls
```

lifespan

Definition at line 126 of file tensor_wrap_specs.h.

8.379.3.5 name

```
std::string nntrainer::TensorSpecV2::name
```

Identifier

Definition at line 124 of file tensor_wrap_specs.h.

8.379.3.6 offset

```
unsigned int nntrainer::TensorSpecV2::offset = 0u
```

ONLY USED FOR READ_ONLY_VIEW, MAYBE_MODIFYING_VIEW tensor offset

Definition at line 131 of file tensor_wrap_specs.h.

8.379.3.7 reference_name

```
std::string nntrainer::TensorSpecV2::reference_name
```

reference name

Definition at line 132 of file tensor_wrap_specs.h.

8.379.3.8 request_type

```
RequestType nntrainer::TensorSpecV2::request_type = RequestType::UNIQUE
```

Type of request

Definition at line 123 of file tensor_wrap_specs.h.

The documentation for this struct was generated from the following file:

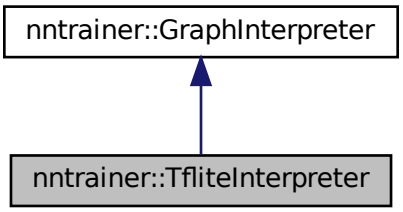
- tensor/tensor_wrap_specs.h

8.380 nntrainer::TfliteInterpreter Class Reference

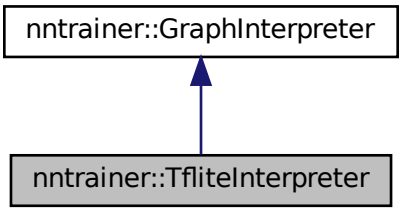
tflite graph interpreter class

`#include <tflite_interpreter.h>`

Inheritance diagram for nntrainer::TfliteInterpreter:



Collaboration diagram for nntrainer::TfliteInterpreter:



Public Member Functions

- TfliteInterpreter (const AppContext &app_context_=AppContext::Global())

  Construct a new tflite Graph Interpreter object.
- virtual ~TfliteInterpreter ()=default

  Destroy the Tflite Interpreter object.
- void serialize (const GraphRepresentation &representation, const std::string &out) override
- GraphRepresentation deserialize (const std::string &in) override

  deserialize graph from a stream

8.380.1 Detailed Description

tflite graph interpreter class

Definition at line 24 of file tflite_interpreter.h.

8.380.2 Constructor & Destructor Documentation

8.380.2.1 TfliteInterpreter()

```
nntrainer::TfliteInterpreter::TfliteInterpreter (
            const AppContext & app_context_ = AppContext::Global() )  [inline]
```

Construct a new tflite Graph Interpreter object.

Parameters

*app_↩ context_*	app context to create layers

Definition at line 31 of file tflite_interpreter.h.

8.380.2.2 ~TfliteInterpreter()

```
virtual nntrainer::TfliteInterpreter::~TfliteInterpreter ( )  [virtual], [default]
```

Destroy the Tflite Interpreter object.

8.380.3 Member Function Documentation

8.380.3.1 deserialize()

```
GraphRepresentation nntrainer::TfliteInterpreter::deserialize (
            const std::string & in )  [override], [virtual]
```

deserialize graph from a stream

Parameters

in	input file name

Returns

GraphRepresentation graph representation

======== list of things to consider ======== we need to reconstruct some properties from the shape eg) units are not saved as a property

NYI!

Implements nntrainer::GraphInterpreter.

Definition at line 664 of file tflite_interpreter.cpp.

8.380.3.2 serialize()

```
void nntrainer::TfliteInterpreter::serialize (
            const GraphRepresentation & representation,
            const std::string & out )  [override], [virtual]
```

Todo check if graph is finalized & initialized and ready to serialize.

0. remove batch normalization layer in GraphRepresentation

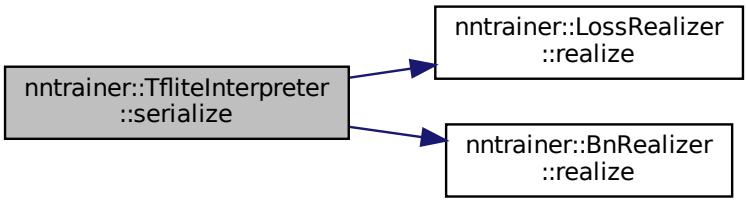
1. remove loss layer in GraphRepresentation
2. The graph must have weights, input dims, output dims set

build TfOpIdxMap from opNodes

Implements nntrainer::GraphInterpreter.

Definition at line 628 of file tflite_interpreter.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- compiler/tflite_interpreter.h
- compiler/tflite_interpreter.cpp

8.381 nntrainer::TfOpNode Class Reference

tensorflow operational node representation. This class contains, information to build operation flatbuffer

`#include <tflite_opnode.h>`

Public Types

- using **Variables** = std::vector< const Tensor * >
- using **TransformFn** = std::function< std::vector< Tensor >(std::vector< const Tensor * > &)>

Public Member Functions

- TfOpNode ()

  Construct a new Tf object.
- void finalize ()

  finalize tf op node will be transformed to required variables in this phase, weights are merged into inputs
- void setLayerNode (const LayerNode &layer)

  Set common informations from layer node.
- void setWeightTransformFn (TransformFn fn)

  Set the Weight Transform Fn object.
- void setInputTransformFn (TransformFn fn)

  Set the Input Transform Fn object.
- void setOpType (tflite::BuiltinOperator op_type_)

  Set the Op Type object.
- void setBuiltinOptions (tflite::BuiltinOptions builtin_option_type_, const flatbuffers::Offset< void > &builtin_ops_)

  Set the Builtin Options object,.
- void setNeedReorderWeight ()

  Set the Need Reorder Weight object.
- void setTrainable (bool trainable)

  Set the Trainable object.
- void weightReorder (unsigned int node_count)

  Reorder Weight in case of NCHW --> NHWC.
- Variables & getInputs ()

  Get the Inputs object.
- const Variables & getWeights () const

  Get the weights object.
- Variables & getWeights ()

  Get the weights object.
- const Variables & getInputs () const

  Get the Inputs object.
- Variables & getOutputs ()

  Get the Outputs object.
- const Variables & getOutputs () const

  Get the Outputs object.
- bool isInputNode () const

  check if this op node is model input
- bool isOutputNode () const

check if this op node is model output

- bool isVirtualNode () const

  check if this op node is virtual node

- bool isNeedReorder () const

  check if this layer need to reorder

- bool isTrainable () const

  check if this layer is trainable

- const tflite::BuiltinOperator getOpType () const

  Get the Op Type object.

- const tflite::BuiltinOptions getOptionType () const

  Get the Op Type object.

- flatbuffers::Offset< void > getBuiltinOps () const

  Get the Op Options object.

- const std::vector< TfOpNode * > & getInputNodes () const

  Get input nodes.

- void arity (size_t value)

  Set arity.

- const unsigned arity () const

  Get arity.

- void setArg (size_t index, TfOpNode *node)

  Set n-th argument of the node.

- TfOpNode * arg (size_t index) const

  Get n-th argument of the node.

8.381.1 Detailed Description

tensorflow operational node representation. This class contains, information to build operation flatbuffer

Definition at line 33 of file tflite_opnode.h.

8.381.2 Constructor & Destructor Documentation

8.381.2.1 TfOpNode()

```
nntrainer::TfOpNode::TfOpNode ( )
```

Construct a new Tf object.

Definition at line 22 of file tflite_opnode.cpp.

8.381.3 Member Function Documentation

8.381.3.1 arg()

```
TfOpNode* nntrainer::TfOpNode::arg (
            size_t index ) const  [inline]
```

Get n-th argument of the node.

Returns

TfOpNode *input_nodes.at(index)

Definition at line 248 of file tflite_opnode.h.

8.381.3.2 arity() [1/2]

```
const unsigned nntrainer::TfOpNode::arity ( ) const  [inline]
```

Get arity.

Returns

const unsigned input_nodes.size()

Definition at line 233 of file tflite_opnode.h.

8.381.3.3 arity() [2/2]

```
void nntrainer::TfOpNode::arity (
            size_t value )  [inline]
```

Set arity.

Parameters

value	value to set

Definition at line 226 of file tflite_opnode.h.

8.381.3.4 finalize()

```
void nntrainer::TfOpNode::finalize ( )
```

finalize tf op node will be transformed to required variables in this phase, weights are merged into inputs

basically, result.size() == v.size() except InputLayer because a Transpose operator is added for converting nchw to nhwc

Todo comment out below codes. TfOpNode needs to have LayerNode pointer

Definition at line 177 of file tflite_opnode.cpp.

8.381.3.5 getBuiltinOps()

```
flatbuffers::Offset< void > nntrainer::TfOpNode::getBuiltinOps ( ) const
```

Get the Op Options object.

Parameters

f	Flatbuffer Builder

Return values

const	tflite::Offset<void>

Definition at line 201 of file tflite_opnode.cpp.

8.381.3.6 getInputNodes()

```
const std::vector<TfOpNode *>& nntrainer::TfOpNode::getInputNodes ( ) const  [inline]
```

Get input nodes.

Returns

const std::vector<TfOpNode *> &input_nodes

Definition at line 219 of file tflite_opnode.h.

8.381.3.7 getInputs() [1/2]

```
Variables& nntrainer::TfOpNode::getInputs ( )  [inline]
```

Get the Inputs object.

Returns

Variables& inputs

Definition at line 115 of file tflite_opnode.h.

8.381.3.8 getInputs() [2/2]

```
const Variables& nntrainer::TfOpNode::getInputs ( ) const  [inline]
```

Get the Inputs object.

Returns

const Variables& inputs

Definition at line 136 of file tflite_opnode.h.

8.381.3.9 getOptionType()

```
const tflite::BuiltinOptions nntrainer::TfOpNode::getOptionType ( ) const  [inline]
```

Get the Op Type object.

Returns

const tflite::BuiltinOperator

Definition at line 203 of file tflite_opnode.h.

8.381.3.10 getOpType()

```
const tflite::BuiltinOperator nntrainer::TfOpNode::getOpType ( ) const  [inline]
```

Get the Op Type object.

Returns

const tflite::BuiltinOperator

Definition at line 196 of file tflite_opnode.h.

8.381.3.11 getOutputs() [1/2]

```
Variables& nntrainer::TfOpNode::getOutputs ( )  [inline]
```

Get the Outputs object.

Returns

Variables&

Definition at line 143 of file tflite_opnode.h.

8.381.3.12 getOutputs() [2/2]

```
const Variables& nntrainer::TfOpNode::getOutputs ( ) const  [inline]
```

Get the Outputs object.

Returns

const Variables& outputs

Definition at line 150 of file tflite_opnode.h.

8.381.3.13 getWeights() [1/2]

```
Variables& nntrainer::TfOpNode::getWeights ( )  [inline]
```

Get the weights object.

Returns

Variables& weights

Definition at line 129 of file tflite_opnode.h.

8.381.3.14 getWeights() [2/2]

```
const Variables& nntrainer::TfOpNode::getWeights ( ) const  [inline]
```

Get the weights object.

Returns

const Variables& weights

Definition at line 122 of file tflite_opnode.h.

8.381.3.15 isInputNode()

```
bool nntrainer::TfOpNode::isInputNode ( ) const  [inline]
```

check if this op node is model input

Return values

true	if op node is model input
false	if op node is not model input

Definition at line 158 of file tflite_opnode.h.

8.381.3.16 isNeedReorder()

```
bool nntrainer::TfOpNode::isNeedReorder ( ) const  [inline]
```

check if this layer need to reorder

Returns

true if weight need to reorder

false if reordering is not required

Definition at line 181 of file tflite_opnode.h.

8.381.3.17 isOutputNode()

```
bool nntrainer::TfOpNode::isOutputNode ( ) const  [inline]
```

check if this op node is model output

Return values

true	if op node is model output
false	if op node is not model output

Definition at line 166 of file tflite_opnode.h.

8.381.3.18 isTrainable()

```
bool nntrainer::TfOpNode::isTrainable ( ) const  [inline]
```

check if this layer is trainable

Returns

true if layer(OpNode) trainable

false if layer(OpNode) non-trainable

Definition at line 189 of file tflite_opnode.h.

8.381.3.19 isVirtualNode()

```
bool nntrainer::TfOpNode::isVirtualNode ( ) const  [inline]
```

check if this op node is virtual node

virtual node is a node that will not be exported

Definition at line 173 of file tflite_opnode.h.

8.381.3.20 setArg()

```
void nntrainer::TfOpNode::setArg (
            size_t index,
            TfOpNode * node )  [inline]
```

Set n-th argument of the node.

Parameters

index	argument index to set
node	the node to be argument

Definition at line 241 of file tflite_opnode.h.

8.381.3.21 setBuiltinOptions()

```
void nntrainer::TfOpNode::setBuiltinOptions (
            tflite::BuiltinOptions builtin_option_type_,
            const flatbuffers::Offset< void > & builtin_ops_ )
```

Set the Builtin Options object,.

Note

this can go private, export from a layer and fill this out

Parameters

*builtin_option_type_*	builtin option type
*builtin_ops_*	flatbuffer offset of builtin_ops

Definition at line 218 of file tflite_opnode.cpp.

8.381.3.22 setInputTransformFn()

```
void nntrainer::TfOpNode::setInputTransformFn (
            TransformFn fn )
```

Set the Input Transform Fn object.

Parameters

fn	fn will be called before get

Definition at line 118 of file tflite_opnode.cpp.

8.381.3.23 setLayerNode()

```
void nntrainer::TfOpNode::setLayerNode (
            const LayerNode & layer )
```

Set common informations from layer node.

Parameters

layer	node layer node

Todo support more loss layers

set to graph output node if output connection of the node includes loss layer string

Note

this is workaround because it cannot be guaranteed that a loss layer always has a loss type in its name.

There are two ways to pass `GraphRepresentation` parameters to `serialize` method.

1. with loss layer at the end of the graph
2. without loss layer but last node has loss layer output connection

Loss layer of the first case is removed by `LossRealizer` and the previous layer of the loss layer is set as the output node. And, the below logic is for the second case.

assume that loss layers have single output

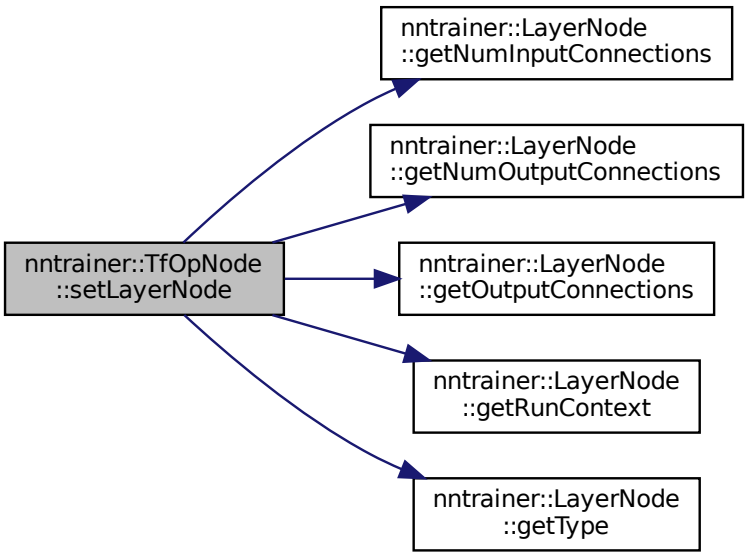
Todo support more virtual nodes

Q1) Why convert from NCHW to NHWC? A1) the tflite uses NHWC format; nntrainer uses NCHW

Q2) Why are only output tensors reshaped? A2) the tflite needs only one tensor between nodes. Therefore, basically, outputs are used for tflite tensors

Definition at line 38 of file tflite_opnode.cpp.

Here is the call graph for this function:



8.381.3.24 setNeedReorderWeight()

```
void nntrainer::TfOpNode::setNeedReorderWeight ( )  [inline]
```

Set the Need Reorder Weight object.

Definition at line 95 of file tflite_opnode.h.

8.381.3.25 setOpType()

```
void nntrainer::TfOpNode::setOpType (
            tflite::BuiltinOperator op_type_ )  [inline]
```

Set the Op Type object.

Parameters

*op_↩ type_*	operation type

Definition at line 79 of file tflite_opnode.h.

8.381.3.26 setTrainable()

```
void nntrainer::TfOpNode::setTrainable (
             bool trainable )  [inline]
```

Set the Trainable object.

Definition at line 101 of file tflite_opnode.h.

8.381.3.27 setWeightTransformFn()

```
void nntrainer::TfOpNode::setWeightTransformFn (
             TransformFn fn )
```

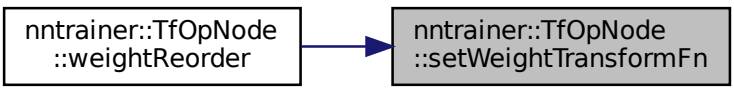
Set the Weight Transform Fn object.

Parameters

fn	fn will be called before get

Definition at line 116 of file tflite_opnode.cpp.

Here is the caller graph for this function:



8.381.3.28 weightReorder()

```
void nntrainer::TfOpNode::weightReorder (
            unsigned int node_count )
```

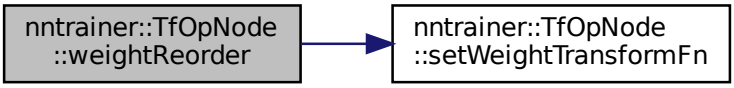
Reorder Weight in case of NCHW --> NHWC.

Parameters

node_count	

Definition at line 120 of file tflite_opnode.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- compiler/tflite_opnode.h
- compiler/tflite_opnode.cpp

8.382 tflite::TileOptionsBuilder Struct Reference

Public Types

- typedef TileOptions **Table**

Public Member Functions

- **TileOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- TileOptionsBuilder & **operator=** (const TileOptionsBuilder &)
- flatbuffers::Offset< TileOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.382.1 Detailed Description

Definition at line 5132 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

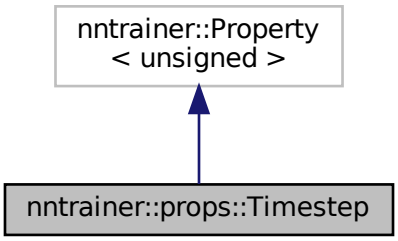
- compiler/tf_schema_generated.h

8.383 nntrainer::props::Timestep Class Reference

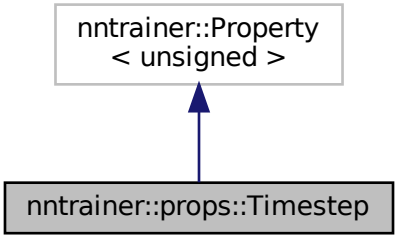
timestep property, timestep is used to identify for which timestep should the lstm/gru/rnn layer do the operation for

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Timestep:



Collaboration diagram for nntrainer::props::Timestep:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key = "timestep"

8.383.1 Detailed Description

timestep property, timestep is used to identify for which timestep should the lstm/gru/rnn layer do the operation for

Definition at line 1105 of file common_properties.h.

8.383.2 Member Typedef Documentation

8.383.2.1 prop_tag

```
using nntrainer::props::Timestep::prop_tag = uint_prop_tag
```

property type

Definition at line 1108 of file common_properties.h.

8.383.3 Member Data Documentation

8.383.3.1 key

```
constexpr const char* nntrainer::props::Timestep::key = "timestep"  [static], [constexpr]
```

unique key to access

Definition at line 1107 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.384 tflite::TopKV2OptionsBuilder Struct Reference

Public Types

- typedef TopKV2Options **Table**

Public Member Functions

- **TopKV2OptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- TopKV2OptionsBuilder & **operator=** (const TopKV2OptionsBuilder &)
- flatbuffers::Offset< TopKV2Options > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.384.1 Detailed Description

Definition at line 4526 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

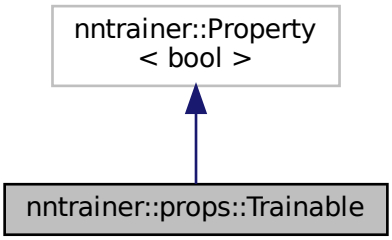
- compiler/tf_schema_generated.h

8.385 nntrainer::props::Trainable Class Reference

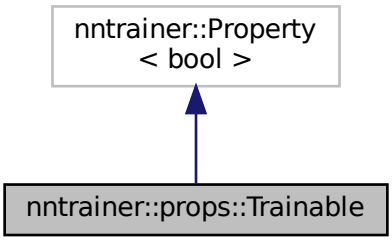
trainable property, use this to set and check how if certain layer is trainable

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Trainable:



Collaboration diagram for nntrainer::props::Trainable:



Public Types

- using **prop_tag** = bool_prop_tag

Public Member Functions

- Trainable (bool val=true)

  Construct a new Trainable object.

Static Public Attributes

- static constexpr const char * **key** = "trainable"

8.385.1 Detailed Description

trainable property, use this to set and check how if certain layer is trainable

Definition at line 99 of file common_properties.h.

8.385.2 Constructor & Destructor Documentation

8.385.2.1 Trainable()

```
nntrainer::props::Trainable::Trainable (
            bool val = true )  [inline]
```

Construct a new Trainable object.

Definition at line 105 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.386 tflite::TransposeConvOptionsBuilder Struct Reference

Public Types

- typedef TransposeConvOptions **Table**

Public Member Functions

- void **add_padding** (tflite::Padding padding)
- void **add_stride_w** (int32_t stride_w)
- void **add_stride_h** (int32_t stride_h)
- **TransposeConvOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- TransposeConvOptionsBuilder & **operator=** (const TransposeConvOptionsBuilder &)
- flatbuffers::Offset< TransposeConvOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.386.1 Detailed Description

Definition at line 5473 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.387 tflite::TransposeOptionsBuilder Struct Reference

Public Types

- typedef TransposeOptions **Table**

Public Member Functions

- **TransposeOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- TransposeOptionsBuilder & **operator=** (const TransposeOptionsBuilder &)
- flatbuffers::Offset< TransposeOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.387.1 Detailed Description

Definition at line 4640 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.388 tflite::Uint16VectorBuilder Struct Reference

Public Types

- typedef Uint16Vector **Table**

Public Member Functions

- void **add_values** (flatbuffers::Offset< flatbuffers::Vector< uint16_t >> values)
- **Uint16VectorBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- Uint16VectorBuilder & **operator=** (const Uint16VectorBuilder &)
- flatbuffers::Offset< Uint16Vector > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.388.1 Detailed Description

Definition at line 2225 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.389 tflite::Uint8VectorBuilder Struct Reference

Public Types

- typedef Uint8Vector **Table**

Public Member Functions

- void **add_values** (flatbuffers::Offset< flatbuffers::Vector< uint8_t >> values)
- **Uint8VectorBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- Uint8VectorBuilder & **operator=** (const Uint8VectorBuilder &)
- flatbuffers::Offset< Uint8Vector > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.389.1 Detailed Description

Definition at line 2278 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.390 tflite::UnidirectionalSequenceLSTMOptionsBuilder Struct Reference

Public Types

- typedef UnidirectionalSequenceLSTMOptions **Table**

Public Member Functions

- void **add_fused_activation_function** (tflite::ActivationFunctionType fused_activation_function)
- void **add_cell_clip** (float cell_clip)
- void **add_proj_clip** (float proj_clip)
- void **add_time_major** (bool time_major)
- void **add_asymmetric_quantize_inputs** (bool asymmetric_quantize_inputs)
- **UnidirectionalSequenceLSTMOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- UnidirectionalSequenceLSTMOptionsBuilder & **operator=** (const UnidirectionalSequenceLSTMOptionsBuilder &)
- flatbuffers::Offset< UnidirectionalSequenceLSTMOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.390.1 Detailed Description

Definition at line 3821 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.391 tflite::UniqueOptionsBuilder Struct Reference

Public Types

- typedef UniqueOptions **Table**

Public Member Functions

- void **add_idx_out_type** (tflite::TensorType idx_out_type)
- **UniqueOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- UniqueOptionsBuilder & **operator=** (const UniqueOptionsBuilder &)
- flatbuffers::Offset< UniqueOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.391.1 Detailed Description

Definition at line 6421 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

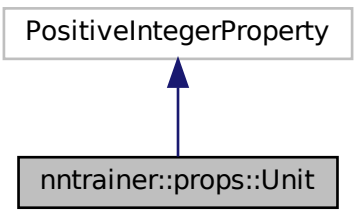
- compiler/tf_schema_generated.h

8.392 nntrainer::props::Unit Class Reference

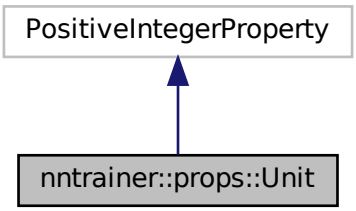
unit property, unit is used to measure how many weights are there

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::Unit:



Collaboration diagram for nntrainer::props::Unit:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key = "unit"

8.392.1 Detailed Description

unit property, unit is used to measure how many weights are there

Definition at line 88 of file common_properties.h.

8.392.2 Member Typedef Documentation

8.392.2.1 prop_tag

```
using nntrainer::props::Unit::prop_tag = uint_prop_tag
```

property type

Definition at line 91 of file common_properties.h.

8.392.3 Member Data Documentation

8.392.3.1 key

```
constexpr const char* nntrainer::props::Unit::key = "unit"  [static], [constexpr]
```

unique key to access

Definition at line 90 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.393 tflite::UnpackOptionsBuilder Struct Reference

Public Types

- typedef UnpackOptions **Table**

Public Member Functions

- void **add_num** (int32_t num)
- void **add_axis** (int32_t axis)
- **UnpackOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- UnpackOptionsBuilder & **operator=** (const UnpackOptionsBuilder &)
- flatbuffers::Offset< UnpackOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.393.1 Detailed Description

Definition at line 6080 of file tf_schema_generated.h.

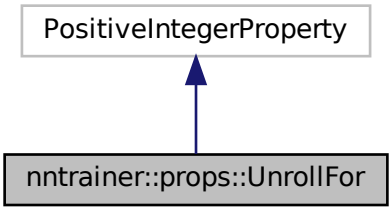
The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

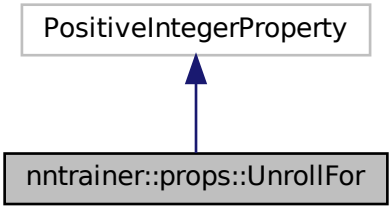
8.394 nntrainer::props::UnrollFor Class Reference

Property check unroll_for.

Inheritance diagram for nntrainer::props::UnrollFor:



Collaboration diagram for nntrainer::props::UnrollFor:



Public Types

- using **prop_tag** = uint_prop_tag

Public Member Functions

- **UnrollFor** (const unsigned &value=1)

Static Public Attributes

- static constexpr const char * **key** = "unroll_for"

8.394.1 Detailed Description

Property check unroll_for.

Definition at line 39 of file recurrent_realizer.cpp.

The documentation for this class was generated from the following file:

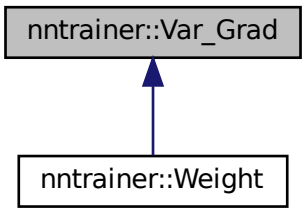
- compiler/recurrent_realizer.cpp

8.395 nntrainer::Var_Grad Class Reference

Variable with Gradient, and its corresponding need_gradient property.

`#include <var_grad.h>`

Inheritance diagram for nntrainer::Var_Grad:



Public Types

- typedef VarGradSpec Spec
  Specification of the Var_Grad.

Public Member Functions

- Var_Grad ()
  Var_Grad default constructor.
- virtual ~Var_Grad ()=default
  Var_Grad default destructor.
- Var_Grad (const TensorDim &dim, const Tensor::Initializer init=Tensor::Initializer::NONE, bool ng=true, bool alloc_now=false, const std::string &name="")
  Construct a new Var_Grad object.
- Var_Grad (const Spec &spec, bool alloc_now=false)
  Construct a new Var_Grad object.
- Var_Grad (const Tensor &v, const Tensor &g, const std::string &n="", bool is_dependent=false)

Construct a new Var_Grad object.

- Var_Grad (Tensor ∗v, Tensor ∗g, bool is_dependent=false)

  Construct a new Var_Grad object.
- Var_Grad (const Var_Grad &rhs)=default

  Copy constructor for Var_Grad.
- Var_Grad (Var_Grad &&rhs)=default

  Move constructor for Var_Grad.
- Var_Grad & operator= (const Var_Grad &rhs)=default

  copy assigment
- Var_Grad & operator= (Var_Grad &&rhs)=default

  move assignment
- virtual void initializeVariable (const Tensor &preallocated=Tensor())

  Initialize the variable.
- virtual void initializeGradient (const Tensor &preallocated=Tensor())

  Initialize the gradient for the variable.
- TensorDim getDim () const

  Get the TensorDim.
- const std::string & getName () const

  Get the name of the variable.
- const std::string & getGradientName () const

  Get the name of the gradient.
- Tensor getVariable () const

  Get the variable tensor.
- Tensor getGradient () const

  Get the Gradient tensor.
- void resetGradient ()

  Reset the gradient to 0.
- void setBatchSize (unsigned int batch)

  Set batch size.
- Tensor & getVariableRef ()

  Get the variable tensor (by reference)
- Tensor & getGradientRef ()

  Get the Gradient tensor (by reference)
- const Tensor & getVariableRef () const

  Get the variable tensor (by reference)
- const Tensor & getGradientRef () const

  Get the Gradient tensor (by reference)
- bool hasGradient () const

  If this variable has gradient.
- bool isDependent () const

  check if given weight is dependent to other weight
- void setAsGradientFirstAccess ()

  Set the As First Gradient Access.
- void setAsGradientLastAccess ()

  Set the As Gradient Last Access object.
- bool isGradientFirstAccess () const

  check if given weight at the last execution order (first access of gradient)
- bool isGradientLastAccess () const

  check if given weight at the first execution order (last access of gradient)
- float getGradientNorm () const

  Get the norm of the gradient.

Static Public Attributes

- static const std::string **grad_suffix** = ":grad"

Protected Attributes

- bool is_dependent
- bool is_first_access_gradient
- bool is_last_access_gradient
- std::shared_ptr< Tensor > var
- std::shared_ptr< Tensor > grad

8.395.1 Detailed Description

Variable with Gradient, and its corresponding need_gradient property.

Definition at line 28 of file var_grad.h.

8.395.2 Member Typedef Documentation

8.395.2.1 Spec

```
typedef VarGradSpec nntrainer::Var_Grad::Spec
```

Specification of the Var_Grad.

The tuple values are dimension, need_gradient property, and the name of the Var_Grad object.

Definition at line 36 of file var_grad.h.

8.395.3 Constructor & Destructor Documentation

8.395.3.1 Var_Grad() [1/7]

```
nntrainer::Var_Grad::Var_Grad ( )  [inline]
```

Var_Grad default constructor.

Note

Default variable is not need_gradient as gradient is 0 dim tensor

Definition at line 42 of file var_grad.h.

8.395.3.2 Var_Grad() [2/7]

```
nntrainer::Var_Grad::Var_Grad (
            const TensorDim & dim,
            const Tensor::Initializer init = Tensor::Initializer::NONE,
            bool ng = true,
            bool alloc_now = false,
            const std::string & name = "" )  [explicit]
```

Construct a new Var_Grad object.

Parameters

dim	Variable and gradient tensor dimension
ng	If the variable is need_gradient
alloc_now	The memory for the var_grad tensors be allocated upon init
name	Name for this Var_Grad

Todo gradient initializer should be none, and then they should be set zero right before using by the user itself.

Definition at line 21 of file var_grad.cpp.

8.395.3.3 Var_Grad() [3/7]

```
nntrainer::Var_Grad::Var_Grad (
            const Spec & spec,
            bool alloc_now = false )  [inline], [explicit]
```

Construct a new Var_Grad object.

Parameters

spec	Var_Grad specification

Definition at line 67 of file var_grad.h.

8.395.3.4 Var_Grad() [4/7]

```
nntrainer::Var_Grad::Var_Grad (
            const Tensor & v,
            const Tensor & g,
            const std::string & n = "",
            bool is_dependent = false )  [inline], [explicit]
```

Construct a new Var_Grad object.

Parameters

v	Already created variable object
g	Already created gradient object
n	Name for this Var_Grad
is_dependent	true if the var grad is dependent

Note

This API is not recommended for usage and must be used for internal uses only, as Var_Grad does not own the tensors v and g, and can go invalid if the owner of these tensors free the tensors.

Definition at line 87 of file var_grad.h.

8.395.3.5 Var_Grad() [5/7]

```
nntrainer::Var_Grad::Var_Grad (
              Tensor * v,
              Tensor * g,
              bool is_dependent = false )  [inline], [explicit]
```

Construct a new Var_Grad object.

Parameters

v	ptr to already created variable tensor
g	ptr to already created gradient tensor
is_dependent	true if the given var grad is dependent

Definition at line 107 of file var_grad.h.

8.395.3.6 Var_Grad() [6/7]

```
nntrainer::Var_Grad::Var_Grad (
              const Var_Grad & rhs )  [default]
```

Copy constructor for Var_Grad.

Parameters

rhs	Var_Grad to construct from

8.395.3.7 Var_Grad() [7/7]

```
nntrainer::Var_Grad::Var_Grad (
              Var_Grad && rhs )  [default]
```

Move constructor for Var_Grad.

Parameters

rhs	Var_Grad to construct from

8.395.4 Member Function Documentation

8.395.4.1 getDim()

```
TensorDim nntrainer::Var_Grad::getDim ( ) const  [inline]
```

Get the TensorDim.

Returns

TensorDim Dimension

Definition at line 166 of file var_grad.h.

8.395.4.2 getGradient()

```
Tensor nntrainer::Var_Grad::getGradient ( ) const  [inline]
```

Get the Gradient tensor.

Returns

Tensor Gradient tensor

Definition at line 194 of file var_grad.h.

8.395.4.3 getGradientName()

```
const std::string& nntrainer::Var_Grad::getGradientName ( ) const  [inline]
```

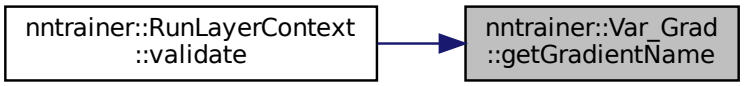
Get the name of the gradient.

Returns

std::string name of the gradient

Definition at line 180 of file var_grad.h.

Here is the caller graph for this function:



8.395.4.4 getGradientNorm()

```
float nntrainer::Var_Grad::getGradientNorm ( ) const  [inline]
```

Get the norm of the gradient.

Returns

float l2 norm of the gradient

Definition at line 299 of file var_grad.h.

8.395.4.5 getGradientRef() [1/2]

```
Tensor& nntrainer::Var_Grad::getGradientRef ( )  [inline]
```

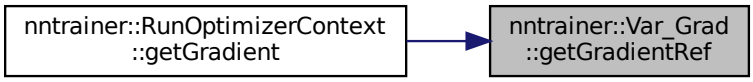
Get the Gradient tensor (by reference)

Returns

Tensor Gradient tensor

Definition at line 228 of file var_grad.h.

Here is the caller graph for this function:



8.395.4.6 getGradientRef() [2/2]

```
const Tensor& nntrainer::Var_Grad::getGradientRef ( ) const  [inline]
```

Get the Gradient tensor (by reference)

Returns

Tensor Gradient tensor

Definition at line 242 of file var_grad.h.

8.395.4.7 getName()

```
const std::string& nntrainer::Var_Grad::getName ( ) const  [inline]
```

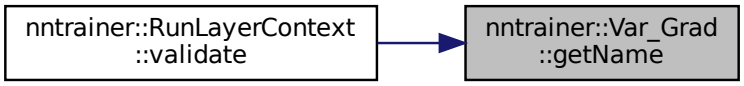
Get the name of the variable.

Returns

std::string name of the variable

Definition at line 173 of file var_grad.h.

Here is the caller graph for this function:



8.395.4.8 getVariable()

```
Tensor nntrainer::Var_Grad::getVariable ( ) const  [inline]
```

Get the variable tensor.

Returns

Tensor Variable tensor

Definition at line 187 of file var_grad.h.

8.395.4.9 getVariableRef() [1/2]

```
Tensor& nntrainer::Var_Grad::getVariableRef ( )  [inline]
```

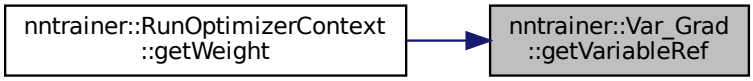
Get the variable tensor (by reference)

Returns

Tensor Variable tensor

Definition at line 221 of file var_grad.h.

Here is the caller graph for this function:



8.395.4.10 getVariableRef() [2/2]

```
const Tensor& nntrainer::Var_Grad::getVariableRef ( ) const  [inline]
```

Get the variable tensor (by reference)

Returns

Tensor Variable tensor

Definition at line 235 of file var_grad.h.

8.395.4.11 hasGradient()

```
bool nntrainer::Var_Grad::hasGradient ( ) const  [inline]
```

If this variable has gradient.

Returns

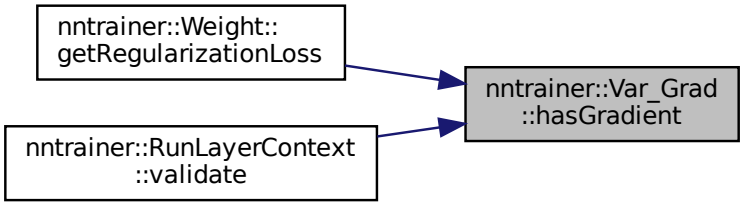
true if the var_grad as gradient set, else false

Note

this is can return is the var_grad needs gradient but it not empty

Definition at line 251 of file var_grad.h.

Here is the caller graph for this function:



8.395.4.12 initializeGradient()

```
void nntrainer::Var_Grad::initializeGradient (
            const Tensor & preallocated = Tensor() )  [virtual]
```

Initialize the gradient for the variable.

Parameters

preallocated	if initialized, use this tensor for gradient memory

Making a new tensor is intentional here as this tensor is not shared with other layers but the internal memory is.

intentionally not initialized tensor memory for shared tensors

Definition at line 50 of file var_grad.cpp.

8.395.4.13 initializeVariable()

```
void nntrainer::Var_Grad::initializeVariable (
            const Tensor & preallocated = Tensor() )  [virtual]
```

Initialize the variable.

Parameters

preallocated	if initialized, use this tensor for variable memory

Making a new tensor is intentional here as this tensor is not shared with other layers but the internal memory is.

intentionally not initialized tensor memory for shared tensors

Definition at line 41 of file var_grad.cpp.

8.395.4.14 isDependent()

```
bool nntrainer::Var_Grad::isDependent ( ) const  [inline]
```

check if given weight is dependent to other weight

Returns

bool return true if the weight is dependent to others

Definition at line 264 of file var_grad.h.

8.395.4.15 isGradientFirstAccess()

```
bool nntrainer::Var_Grad::isGradientFirstAccess ( ) const  [inline]
```

check if given weight at the last execution order (first access of gradient)

Returns

bool true if last access

Definition at line 284 of file var_grad.h.

8.395.4.16 isGradientLastAccess()

```
bool nntrainer::Var_Grad::isGradientLastAccess ( ) const  [inline]
```

check if given weight at the first execution order (last access of gradient)

Returns

bool true if last access

Definition at line 292 of file var_grad.h.

8.395.4.17 operator=() [1/2]

```
Var_Grad& nntrainer::Var_Grad::operator= (
            const Var_Grad & rhs )  [default]
```

copy assigment

Parameters

rhs	copy from

Returns

Var_Grad& Updated Var_Grad

8.395.4.18 operator=() [2/2]

```
Var_Grad& nntrainer::Var_Grad::operator= (
            Var_Grad && rhs ) [default]
```

move assignment

Parameters

rhs	move from

Returns

Var_Grad& Updated Var_Grad

8.395.4.19 resetGradient()

```
void nntrainer::Var_Grad::resetGradient ( ) [inline]
```

Reset the gradient to 0.

zero the gradient

Definition at line 199 of file var_grad.h.

8.395.4.20 setAsGradientFirstAccess()

```
void nntrainer::Var_Grad::setAsGradientFirstAccess ( ) [inline]
```

Set the As First Gradient Access.

Definition at line 270 of file var_grad.h.

8.395.4.21 setAsGradientLastAccess()

```
void nntrainer::Var_Grad::setAsGradientLastAccess ( )  [inline]
```

Set the As Gradient Last Access object.

Definition at line 276 of file var_grad.h.

8.395.4.22 setBatchSize()

```
void nntrainer::Var_Grad::setBatchSize (
            unsigned int batch )  [inline]
```

Set batch size.

Parameters

batch	batch size

Definition at line 209 of file var_grad.h.

8.395.5 Member Data Documentation

8.395.5.1 grad

```
std::shared_ptr<Tensor> nntrainer::Var_Grad::grad  [protected]
```

gradient for the variable

Definition at line 312 of file var_grad.h.

8.395.5.2 is_dependent

```
bool nntrainer::Var_Grad::is_dependent  [protected]
```

check if the weight tensor is burrowed from somewhere thus it is dependent

Definition at line 304 of file var_grad.h.

8.395.5.3 is_first_access_gradient

```
bool nntrainer::Var_Grad::is_first_access_gradient  [protected]
```

check if current weight tensor is first access

Definition at line 306 of file var_grad.h.

8.395.5.4 is_last_access_gradient

```
bool nntrainer::Var_Grad::is_last_access_gradient  [protected]
```

check if current weight tensor is last access

Definition at line 308 of file var_grad.h.

8.395.5.5 var

```
std::shared_ptr<Tensor> nntrainer::Var_Grad::var  [protected]
```

variable to be updated and used

Definition at line 311 of file var_grad.h.

The documentation for this class was generated from the following files:

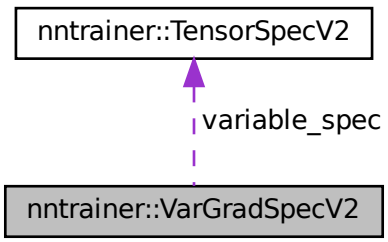
- tensor/var_grad.h
- tensor/var_grad.cpp

8.396 nntrainer::VarGradSpecV2 Struct Reference

variable + gradient specification

```
#include <tensor_wrap_specs.h>
```

Collaboration diagram for nntrainer::VarGradSpecV2:



Public Member Functions

- VarGradSpecV2 ()=default

  Construct a new Var Grad Spec V2 object.
- VarGradSpecV2 (const VarGradSpecV2 &rhs)

  Copy construct.
- VarGradSpecV2 & operator= (const VarGradSpecV2 &rhs)

  copy assignment
- VarGradSpecV2 (VarGradSpecV2 &&) noexcept=default

  Move Construct.
- VarGradSpecV2 & **operator=** (VarGradSpecV2 &&) noexcept=default

Public Attributes

- TensorSpecV2 variable_spec
- std::unique_ptr< TensorSpecV2 > gradient_spec

8.396.1 Detailed Description

variable + gradient specification

Definition at line 143 of file tensor_wrap_specs.h.

8.396.2 Constructor & Destructor Documentation

8.396.2.1 VarGradSpecV2() [1/3]

```
nntrainer::VarGradSpecV2::VarGradSpecV2 ( )  [default]
```

Construct a new Var Grad Spec V2 object.

8.396.2.2 VarGradSpecV2() [2/3]

```
nntrainer::VarGradSpecV2::VarGradSpecV2 (
            const VarGradSpecV2 & rhs )  [inline]
```

Copy construct.

Parameters

rhs	

Definition at line 156 of file tensor_wrap_specs.h.

8.396.2.3 VarGradSpecV2() [3/3]

```
nntrainer::VarGradSpecV2::VarGradSpecV2 (
            VarGradSpecV2 &&  )  [default], [noexcept]
```

Move Construct.

8.396.3 Member Function Documentation

8.396.3.1 operator=()

```
VarGradSpecV2& nntrainer::VarGradSpecV2::operator= (
            const VarGradSpecV2 & rhs )  [inline]
```

copy assignment

Parameters

rhs	

Returns

VarGradSpecV2&

Definition at line 168 of file tensor_wrap_specs.h.

8.396.4 Member Data Documentation

8.396.4.1 gradient_spec

```
std::unique_ptr<TensorSpecV2> nntrainer::VarGradSpecV2::gradient_spec
```

Initial value:

```
=
    nullptr
```

gradient spec, if null it cannot be trained

Definition at line 184 of file tensor_wrap_specs.h.

8.396.4.2 variable_spec

```
TensorSpecV2 nntrainer::VarGradSpecV2::variable_spec
```

variable spec

Definition at line 183 of file tensor_wrap_specs.h.

The documentation for this struct was generated from the following file:

- tensor/tensor_wrap_specs.h

8.397 nntrainer::ViewQueue< T > Class Template Reference

Thread Safe Queue implementation dedicated for the non-owing pointer.

```
#include <iteration_queue.h>
```

Public Member Functions

- ViewQueue ()
  Construct a new queue.
- void push (T *data)
  push data to queue
- T * waitAndPop ()
  pop data from the queue, wait if empty
- bool isEmpty () const
  check if current queue is empty
- std::queue< T * >::size_type size () const
  check if current queue is empty

8.397.1 Detailed Description

template<typename T>
class nntrainer::ViewQueue< T >

Thread Safe Queue implementation dedicated for the non-owing pointer.

Template Parameters

original	type of the view (T * will be pushed and pop)

Definition at line 38 of file iteration_queue.h.

8.397.2 Member Function Documentation

8.397.2.1 isEmpty()

```
template<typename T >
bool nntrainer::ViewQueue< T >::isEmpty ( ) const  [inline]
```

check if current queue is empty

Returns

bool true if empty

Definition at line 79 of file iteration_queue.h.

8.397.2.2 push()

```
template<typename T >
void nntrainer::ViewQueue< T >::push (
            T * data )  [inline]
```

push data to queue

Parameters

data	data to put

Definition at line 50 of file iteration_queue.h.

8.397.2.3 size()

```
template<typename T >
std::queue<T *>::size_type nntrainer::ViewQueue< T >::size ( ) const  [inline]
```

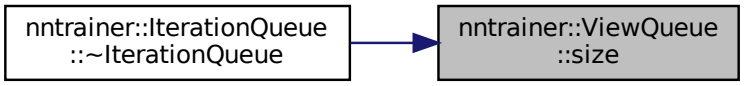
check if current queue is empty

Returns

bool true if empty

Definition at line 89 of file iteration_queue.h.

Here is the caller graph for this function:



8.397.2.4 waitAndPop()

```
template<typename T >
T* nntrainer::ViewQueue< T >::waitAndPop ( )  [inline]
```

pop data from the queue, wait if empty

Note

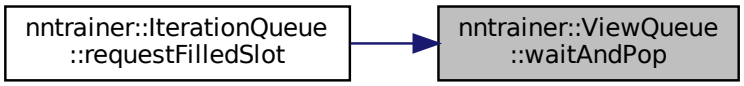
when fail to get, this will return nullptr (eg) when interrupt happens)

Returns

T∗ view of the data

Definition at line 65 of file iteration_queue.h.

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

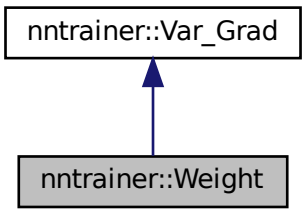
- dataset/iteration_queue.h

8.398 nntrainer::Weight Class Reference

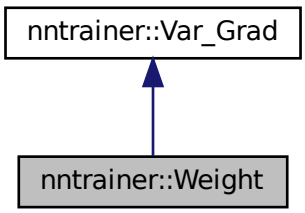
Weight extends over Var_Grad with regularization & optimizer updates.

`#include <weight.h>`

Inheritance diagram for nntrainer::Weight:



Collaboration diagram for nntrainer::Weight:



Public Types

- typedef WeightSpec Spec

  Specification of the Weight.

Public Member Functions

- Weight ()

  Weight default constructor.
- Weight (const TensorDim &dim, const Tensor::Initializer init=Tensor::Initializer::XAVIER_UNIFORM, const WeightRegularizer reg=WeightRegularizer::NONE, const float reg_const=1.0f, const float decay=0.0f, const float clip_by_global_norm=0.0f, bool ng=true, bool alloc_now=false, std::string name="")

  Construct a new Weight object.

- Weight (const Spec &spec, bool alloc_now=false)

  Construct a new Weight object.
- Weight (const Tensor &v, const Tensor &g, const std::string &n="", bool is_dependent=false)

  Construct a new Weight object.
- Weight (Tensor ∗v, Tensor ∗g, const WeightRegularizer reg, const float reg_const, const float decay, bool is_dependent=false, const float max_norm=0.0f)

  Construct a new Weight object.
- Weight (const Weight &rhs)=default

  Copy constructor for weight.
- Weight (Weight &&rhs)=default

  Move constructor for weight.
- Weight & operator= (const Weight &rhs)=default

  copy assigment
- Weight & operator= (Weight &&rhs)=default

  move assignment
- Weight clone () const

  Clone the currnet object.
- void clearOptimizerVariables ()

  Clear optimizer variables.
- void setOptimizerVariables (std::vector< Tensor ∗ > tensors)

  Add optimizer variables.
- Tensor & getOptimizerVariableRef (unsigned int idx)

  Get optimizer variable reference.
- int getNumOptVariable ()

  Get number of optimizer variable.
- bool isWeightRegularizerL2Norm ()

  check if weight regularizer type is l2norm
- bool isWeightDecay ()

  check if weight decay is enabled
- float getRegularizationLoss ()

  Get loss from the regularization of the weight.
- void calcRegularizationGradient ()

  Calculate gradient from the regularization of the weight.
- void calcWeightDecayGradient ()

  Calculate gradient from the decay of the weight.
- void applyGradient (double lr)

  Apply the gradient to the weight.
- bool isGradientClipByGlobalNorm () const

  Check if the gradient is supposed to be clipped by global norm.
- void clipGradientByGlobalNorm (const float global_norm)

  clip the gradient value based on the given global norm

Static Public Member Functions

- static bool isGradientClipByGlobalNorm (const float max_norm)

  Check if the gradient is supposed to be clipped by global norm with the given max_norm value.

Friends

- void swap (Weight &lhs, Weight &rhs) noexcept

  Swap for weight.

Additional Inherited Members

8.398.1 Detailed Description

Weight extends over Var_Grad with regularization & optimizer updates.

Definition at line 29 of file weight.h.

8.398.2 Member Typedef Documentation

8.398.2.1 Spec

```
typedef WeightSpec nntrainer::Weight::Spec
```

Specification of the Weight.

The tuple values are dimension, initializer, regularizer, regularizer_constant, need_gradient property amd name of the Weight object.

Definition at line 37 of file weight.h.

8.398.3 Constructor & Destructor Documentation

8.398.3.1 Weight() [1/6]

```
nntrainer::Weight::Weight (
            const TensorDim & dim,
            const Tensor::Initializer init = Tensor::Initializer::XAVIER_UNIFORM,
            const WeightRegularizer reg = WeightRegularizer::NONE,
            const float reg_const = 1.0f,
            const float decay = 0.0f,
            const float clip_by_global_norm = 0.0f,
            bool ng = true,
            bool alloc_now = false,
            std::string name = "" )  [explicit]
```

Construct a new Weight object.

Parameters

dim	Variable and gradient tensor dimension
init	Initializer for the weight
reg	Regularizer for the weight
reg_const	Constant multiplier for regularizer
ng	If the variable needs gradient
alloc_now	The memory for the weight tensors be allocated upon init
name	Name for this weight

Definition at line 21 of file weight.cpp.

8.398.3.2 Weight() [2/6]

```
nntrainer::Weight::Weight (
            const Spec & spec,
            bool alloc_now = false )  [inline], [explicit]
```

Construct a new Weight object.

Parameters

spec	Weight specification

Definition at line 73 of file weight.h.

8.398.3.3 Weight() [3/6]

```
nntrainer::Weight::Weight (
            const Tensor & v,
            const Tensor & g,
            const std::string & n = "",
            bool is_dependent = false )  [inline], [explicit]
```

Construct a new Weight object.

Parameters

v	Already created variable object
g	Already created gradient object
n	Name for this Weight

Note

This is primarily used to created wrapper of variable extracted from context. If needed, add support for regularizer, and opt_vars.

This API is not recommended for usage and must be used for internal uses only, as Weight does not own the tensors v and g, and can go invalid if the owner of these tensors free the tensors.

Definition at line 99 of file weight.h.

8.398.3.4 Weight() [4/6]

```
nntrainer::Weight::Weight (
             Tensor * v,
             Tensor * g,
             const WeightRegularizer reg,
             const float reg_const,
             const float decay,
             bool is_dependent = false,
             const float max_norm = 0.0f )  [inline], [explicit]
```

Construct a new Weight object.

Parameters

v	ptr to already created variable tensor
g	ptr to already created gradient tensor
reg	Regularizer for the weight
reg_const	Constant multiplier for regularizer

Definition at line 115 of file weight.h.

8.398.3.5 Weight() [5/6]

```
nntrainer::Weight::Weight (
             const Weight & rhs )  [default]
```

Copy constructor for weight.

Parameters

rhs	weight to construct from

8.398.3.6 Weight() [6/6]

```
nntrainer::Weight::Weight (
             Weight && rhs )  [default]
```

Move constructor for weight.

Parameters

rhs	weight to construct from

8.398.4 Member Function Documentation

8.398.4.1 clipGradientByGlobalNorm()

```
void nntrainer::Weight::clipGradientByGlobalNorm (
            const float global_norm )  [inline]
```

clip the gradient value based on the given global norm

Parameters

global_norm	the global norm for all the weights

Definition at line 284 of file weight.h.

8.398.4.2 clone()

```
Weight nntrainer::Weight::clone ( ) const  [inline]
```

Clone the currnet object.

Returns

Cloned copy

Definition at line 176 of file weight.h.

8.398.4.3 getNumOptVariable()

```
int nntrainer::Weight::getNumOptVariable ( )  [inline]
```

Get number of optimizer variable.

Return values

number	of optimizer variable

Definition at line 210 of file weight.h.

8.398.4.4 getOptimizerVariableRef()

```
Tensor& nntrainer::Weight::getOptimizerVariableRef (
            unsigned int idx )  [inline]
```

Get optimizer variable reference.

Parameters

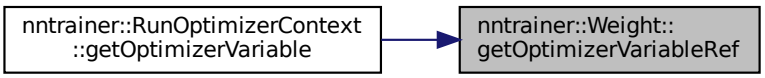
idx	Index of the optimizer variable to get

Return values

Reference	of the optimizer variable

Definition at line 204 of file weight.h.

Here is the caller graph for this function:



8.398.4.5 isGradientClipByGlobalNorm() [1/2]

```
bool nntrainer::Weight::isGradientClipByGlobalNorm ( ) const  [inline]
```

Check if the gradient is supposed to be clipped by global norm.

Returns

true if it is to be clipped

false otherwise

Definition at line 275 of file weight.h.

8.398.4.6 isGradientClipByGlobalNorm() [2/2]

```
static bool nntrainer::Weight::isGradientClipByGlobalNorm (
            const float max_norm )  [inline], [static]
```

Check if the gradient is supposed to be clipped by global norm with the given max_norm value.

Parameters

max_norm	

Returns

true if it is to be clipped

false otherwise

Definition at line 265 of file weight.h.

8.398.4.7 isWeightDecay()

```
bool nntrainer::Weight::isWeightDecay ( )  [inline]
```

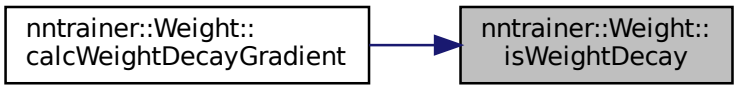
check if weight decay is enabled

Returns

true if weight decay is enabled else false

Definition at line 224 of file weight.h.

Here is the caller graph for this function:



8.398.4.8 isWeightRegularizerL2Norm()

```
bool nntrainer::Weight::isWeightRegularizerL2Norm ( )  [inline]
```

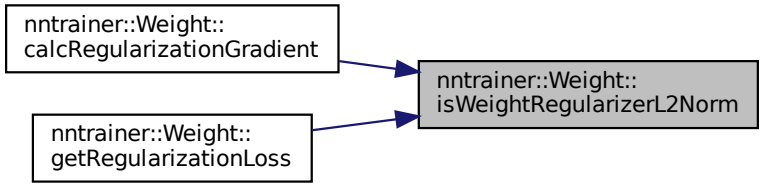
check if weight regularizer type is l2norm

Returns

bool is weight regrulatizer type is L2 Norm

Definition at line 216 of file weight.h.

Here is the caller graph for this function:



8.398.4.9 operator=() [1/2]

```
Weight& nntrainer::Weight::operator= (
            const Weight & rhs )  [default]
```

copy assigment

Parameters

rhs	copy from

Returns

Weight& Updated weight

8.398.4.10 operator=() [2/2]

```
Weight& nntrainer::Weight::operator= (
            Weight && rhs )  [default]
```

move assignment

Parameters

rhs	move from

Returns

Weight& Updated weight

8.398.4.11 setOptimizerVariables()

```
void nntrainer::Weight::setOptimizerVariables (
            std::vector< Tensor * > tensors )  [inline]
```

Add optimizer variables.

Parameters

dim	Optimizer variable dimension

Definition at line 195 of file weight.h.

8.398.5 Friends And Related Function Documentation

8.398.5.1 swap

```
void swap (
            Weight & lhs,
            Weight & rhs )  [friend]
```

Swap for weight.

Parameters

lhs	Swap to
rhs	Swap from

Note

Only swap gradient if need gradient

Definition at line 131 of file weight.h.

The documentation for this class was generated from the following files:

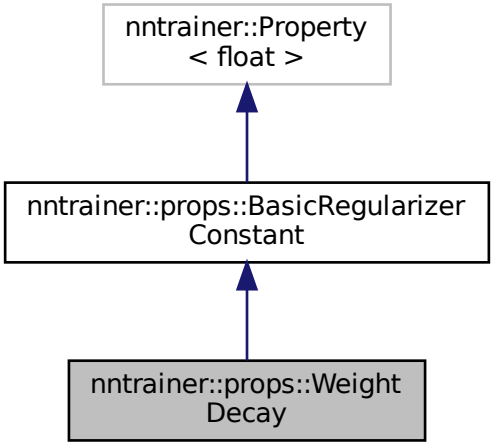
- tensor/weight.h
- tensor/weight.cpp

8.399 nntrainer::props::WeightDecay Class Reference

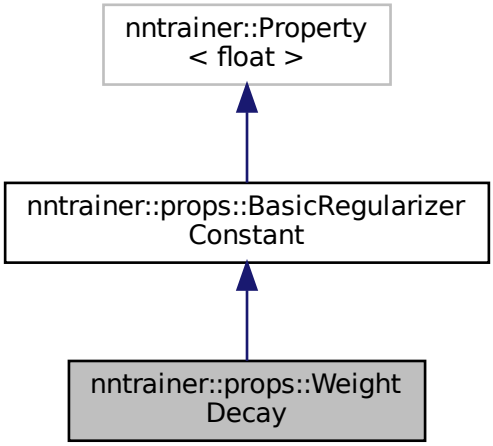
WeightDecay property, this defines how much to decay the weight.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::WeightDecay:



Collaboration diagram for nntrainer::props::WeightDecay:



Public Member Functions

- WeightDecay (float value=0.0f)

  Construct a new WeightDecay object.

Static Public Attributes

- static constexpr const char * key

Additional Inherited Members

8.399.1 Detailed Description

WeightDecay property, this defines how much to decay the weight.

Definition at line 762 of file common_properties.h.

8.399.2 Constructor & Destructor Documentation

8.399.2.1 WeightDecay()

```
nntrainer::props::WeightDecay::WeightDecay (
            float value = 0.0f )
```

Construct a new WeightDecay object.

Definition at line 274 of file common_properties.cpp.

8.399.3 Member Data Documentation

8.399.3.1 key

```
constexpr const char* nntrainer::props::WeightDecay::key  [static], [constexpr]
```

Initial value:

```
=
    "weight_decay"
```

unique key to access

Definition at line 770 of file common_properties.h.

The documentation for this class was generated from the following files:

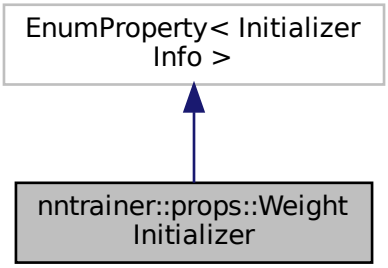
- layers/common_properties.h
- layers/common_properties.cpp

8.400 nntrainer::props::WeightInitializer Class Reference

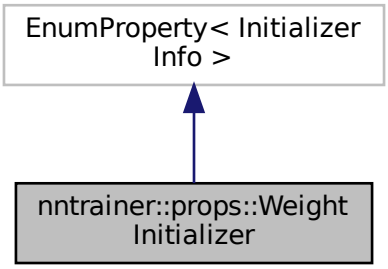
WeightInitializer Initialization Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::WeightInitializer:



Collaboration diagram for nntrainer::props::WeightInitializer:



Public Types

- using **prop_tag** = enum_class_prop_tag

Public Member Functions

- WeightInitializer (Tensor::Initializer value=Tensor::Initializer::XAVIER_UNIFORM)

  Construct a WeightInitializer object.

Static Public Attributes

- static constexpr const char * **key** = "weight_initializer"

8.400.1 Detailed Description

WeightInitializer Initialization Enumeration Information.

Definition at line 915 of file common_properties.h.

The documentation for this class was generated from the following files:

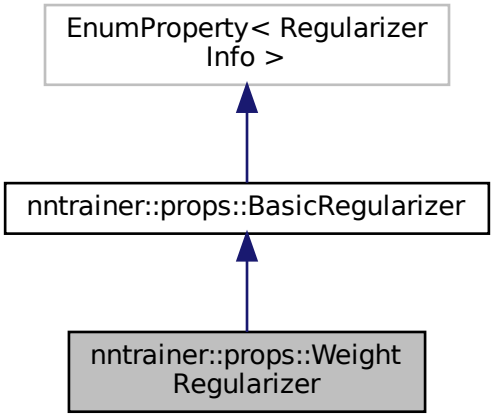
- layers/common_properties.h
- layers/common_properties.cpp

8.401 nntrainer::props::WeightRegularizer Class Reference

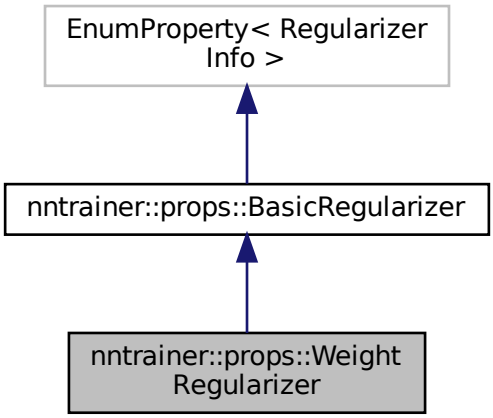
WeightRegularizer Regularization Enumeration Information.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::WeightRegularizer:



Collaboration diagram for nntrainer::props::WeightRegularizer:



Public Member Functions

- WeightRegularizer (nntrainer::WeightRegularizer value=nntrainer::WeightRegularizer::NONE)

  Construct a WeightRegularizer object.

Static Public Attributes

- static constexpr const char * **key** = "weight_regularizer"

Additional Inherited Members

8.401.1 Detailed Description

WeightRegularizer Regularization Enumeration Information.

Definition at line 1034 of file common_properties.h.

The documentation for this class was generated from the following files:

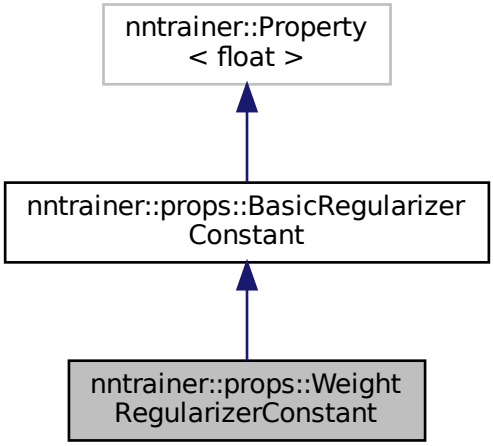
- layers/common_properties.h
- layers/common_properties.cpp

8.402 nntrainer::props::WeightRegularizerConstant Class Reference

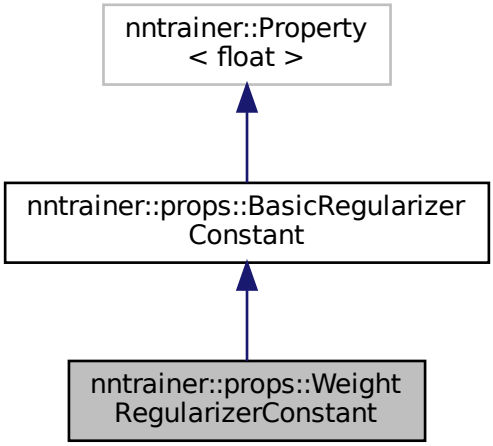
WeightRegularizerConstant property, this defines how much regularize the weight.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::WeightRegularizerConstant:



Collaboration diagram for nntrainer::props::WeightRegularizerConstant:



Public Member Functions

- WeightRegularizerConstant (float value=1.0f)

  Construct a new WeightRegularizerConstant object.

Static Public Attributes

- static constexpr const char * key

Additional Inherited Members

8.402.1 Detailed Description

WeightRegularizerConstant property, this defines how much regularize the weight.

Definition at line 745 of file common_properties.h.

8.402.2 Constructor & Destructor Documentation

8.402.2.1 WeightRegularizerConstant()

```
nntrainer::props::WeightRegularizerConstant::WeightRegularizerConstant (
            float value = 1.0f )
```

Construct a new WeightRegularizerConstant object.

Definition at line 272 of file common_properties.cpp.

8.402.3 Member Data Documentation

8.402.3.1 key

```
constexpr const char* nntrainer::props::WeightRegularizerConstant::key  [static], [constexpr]
```

Initial value:

```
=
    "weight_regularizer_constant"
```

unique key to access

Definition at line 753 of file common_properties.h.

The documentation for this class was generated from the following files:

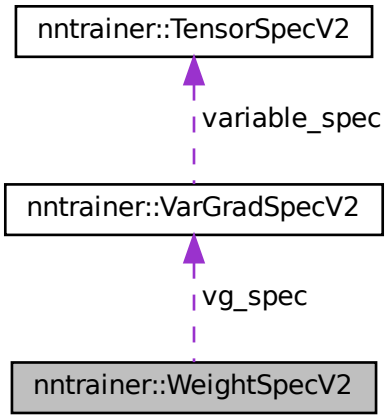
- layers/common_properties.h
- layers/common_properties.cpp

8.403 nntrainer::WeightSpecV2 Struct Reference

weight specification

`#include <tensor_wrap_specs.h>`

Collaboration diagram for nntrainer::WeightSpecV2:



Public Attributes

- VarGradSpecV2 vg_spec
- WeightRegularizer regularizer = WeightRegularizer::NONE
- float regularizer_constant = 0.0f
- float decay = 0.0f
- float clip_by_global_norm = 0.0f

8.403.1 Detailed Description

weight specification

Definition at line 192 of file tensor_wrap_specs.h.

8.403.2 Member Data Documentation

8.403.2.1 clip_by_global_norm

```
float nntrainer::WeightSpecV2::clip_by_global_norm = 0.0f
```

clip the gradient by norm

Definition at line 197 of file tensor_wrap_specs.h.

8.403.2.2 decay

```
float nntrainer::WeightSpecV2::decay = 0.0f
```

decay constant

Definition at line 196 of file tensor_wrap_specs.h.

8.403.2.3 regularizer

```
WeightRegularizer nntrainer::WeightSpecV2::regularizer = WeightRegularizer::NONE
```

regularizer

Definition at line 194 of file tensor_wrap_specs.h.

8.403.2.4 regularizer_constant

```
float nntrainer::WeightSpecV2::regularizer_constant = 0.0f
```

regularizer constant

Definition at line 195 of file tensor_wrap_specs.h.

8.403.2.5 vg_spec

```
VarGradSpecV2 nntrainer::WeightSpecV2::vg_spec
```

variable + graident specification

Definition at line 193 of file tensor_wrap_specs.h.

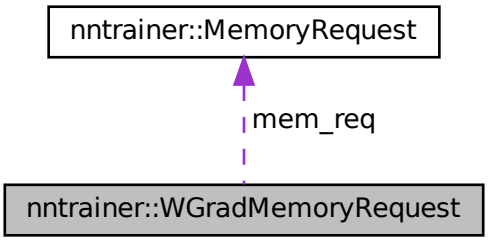
The documentation for this struct was generated from the following file:

- tensor/tensor_wrap_specs.h

8.404 nntrainer::WGradMemoryRequest Struct Reference

Memory Request data structure clubbing for the weight gradient requests.

Collaboration diagram for nntrainer::WGradMemoryRequest:



Public Member Functions

- **WGradMemoryRequest** (MemoryRequest ∗req)

Public Attributes

- MemoryRequest ∗ **mem_req**
- std::vector< std::pair< unsigned int, unsigned int > > **start_end**

8.404.1 Detailed Description

Memory Request data structure clubbing for the weight gradient requests.

Definition at line 54 of file optimized_v2_planner.cpp.

The documentation for this struct was generated from the following file:

- tensor/optimized_v2_planner.cpp

8.405 tflite::WhereOptionsBuilder Struct Reference

Public Types

- typedef WhereOptions **Table**

Public Member Functions

- **WhereOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- WhereOptionsBuilder & **operator=** (const WhereOptionsBuilder &)
- flatbuffers::Offset< WhereOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.405.1 Detailed Description

Definition at line 6546 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

8.406 tflite::WhileOptionsBuilder Struct Reference

Public Types

- typedef WhileOptions **Table**

Public Member Functions

- void **add_cond_subgraph_index** (int32_t cond_subgraph_index)
- void **add_body_subgraph_index** (int32_t body_subgraph_index)
- **WhileOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- WhileOptionsBuilder & **operator=** (const WhileOptionsBuilder &)
- flatbuffers::Offset< WhileOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.406.1 Detailed Description

Definition at line 6782 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

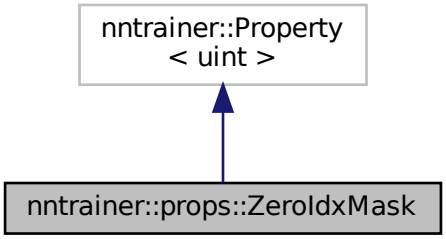
- compiler/tf_schema_generated.h

8.407 nntrainer::props::ZeroIdxMask Class Reference

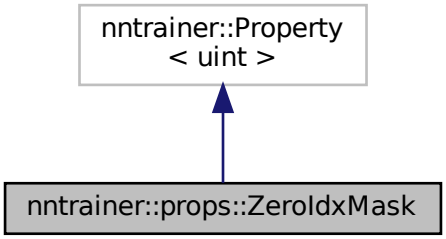
Zero idx mask property for embedding where the value of embedding will be zero.

`#include <common_properties.h>`

Inheritance diagram for nntrainer::props::ZeroIdxMask:



Collaboration diagram for nntrainer::props::ZeroIdxMask:



Public Types

- using prop_tag = uint_prop_tag

Static Public Attributes

- static constexpr const char * key

8.407.1 Detailed Description

Zero idx mask property for embedding where the value of embedding will be zero.

Definition at line 493 of file common_properties.h.

8.407.2 Member Typedef Documentation

8.407.2.1 prop_tag

```
using nntrainer::props::ZeroIdxMask::prop_tag = uint_prop_tag
```

property type

Definition at line 497 of file common_properties.h.

8.407.3 Member Data Documentation

8.407.3.1 key

```
constexpr const char* nntrainer::props::ZeroIdxMask::key  [static], [constexpr]
```

Initial value:

```
=
    "zero_idx_mask"
```

unique key to access

Definition at line 495 of file common_properties.h.

The documentation for this class was generated from the following file:

- layers/common_properties.h

8.408 tflite::ZerosLikeOptionsBuilder Struct Reference

Public Types

- typedef ZerosLikeOptions **Table**

Public Member Functions

- **ZerosLikeOptionsBuilder** (flatbuffers::FlatBufferBuilder &_fbb)
- ZerosLikeOptionsBuilder & **operator=** (const ZerosLikeOptionsBuilder &)
- flatbuffers::Offset< ZerosLikeOptions > **Finish** ()

Public Attributes

- flatbuffers::FlatBufferBuilder & **fbb_**
- flatbuffers::uoffset_t **start_**

8.408.1 Detailed Description

Definition at line 6180 of file tf_schema_generated.h.

The documentation for this struct was generated from the following file:

- compiler/tf_schema_generated.h

Chapter 9

File Documentation

9.1 app_context.cpp File Reference

This file contains app context related functions and classes that manages the global configuration of the current environment.

```
#include <dirent.h>
#include <dlfcn.h>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
#include <iniparser.h>
#include <app_context.h>
#include <layer.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <optimizer.h>
#include <util_func.h>
#include <adam.h>
#include <sgd.h>
#include <activation_layer.h>
#include <addition_layer.h>
#include <attention_layer.h>
#include <bn_layer.h>
#include <centroid_knn.h>
#include <concat_layer.h>
#include <constant_derivative_loss_layer.h>
#include <conv1d_layer.h>
#include <conv2d_layer.h>
#include <cross_entropy_sigmoid_loss_layer.h>
#include <cross_entropy_softmax_loss_layer.h>
#include <dropout.h>
#include <embedding.h>
#include <fc_layer.h>
#include <flatten_layer.h>
#include <gru.h>
#include <grucell.h>
#include <identity_layer.h>
#include <input_layer.h>
```

```
#include <layer_normalization_layer.h>
#include <lr_scheduler_constant.h>
#include <lr_scheduler_exponential.h>
#include <lr_scheduler_step.h>
#include <lstm.h>
#include <lstmcell.h>
#include <mol_attention_layer.h>
#include <mse_loss_layer.h>
#include <multi_head_attention_layer.h>
#include <multiout_layer.h>
#include <permute_layer.h>
#include <plugged_layer.h>
#include <plugged_optimizer.h>
#include <pooling2d_layer.h>
#include <positional_encoding_layer.h>
#include <preprocess_flip_layer.h>
#include <preprocess_l2norm_layer.h>
#include <preprocess_translate_layer.h>
#include <reduce_mean_layer.h>
#include <rnn.h>
#include <rnncell.h>
#include <split_layer.h>
#include <time_dist.h>
#include <zoneout_lstmcell.h>
```

Namespaces

- nntrainer

Functions

- constexpr const char * **getConfPath** ()
- static void nntrainer::fini_global_context_nntrainer (void) __attribute__((destructor))

  finialize global context
- static void nntrainer::add_default_object (AppContext &ac)
- static void **nntrainer::add_extension_object** (AppContext &ac)
- static void **nntrainer::registerer** (AppContext &ac) noexcept
- template<size_t I = 0, typename V , typename... Ts>
  std::enable_if< I==sizeof...(Ts), void >::type nntrainer::parse_properties (V &props, std::tuple< Ts... > &tup)

  base case of iterate_prop, iterate_prop iterates the given tuple

Variables

- static std::string solib_suffix = ".so"

  add #ifdef across platform
- static std::string **layerlib_suffix** = "layer.so"
- static std::string **optimizerlib_suffix** = "optimizer.so"
- static const std::string **func_tag** = "[AppContext] "
- constexpr const char * **DEFAULT_CONF_PATH** = "/etc/nntrainer.ini"
- std::mutex **nntrainer::factory_mutex**
- std::once_flag **nntrainer::global_app_context_init_flag**

- template<size_t I = 0, typename V , typename... Ts>
  std::enable_if< I< sizeof...(Ts), void >::type inline parse_properties(V &props, std::tuple< Ts... > &tup) { std::string name=std::get< I >tup);std::string prop=getConfig(name);if(!prop.empty()) props.↩ push_back(name+"="+prop);parse_properties< I+1 >props, tup);}const std::vector< std::string > App↩ Context::getProperties(void) { std::vector< std::string > properties;auto props=std::tuple("memory_swap", "memory_swap_path");parse_properties(properties, props);return properties;}int AppContext::register↩ Layer(const std::string &library_path, const std::string &base_path) { const std::string full_path=getFull↩ Path(library_path, base_path);void *handle=dlopen(full_path.c_str(), RTLD_LAZY|RTLD_LOCAL);const char *error_msg=dlerror();<< func_tag<< "open plugin failed, reason: "<< error_msg;nntrainer::Layer↩ Pluggable *pluggable=reinterpret_cast< nntrainer::LayerPluggable * > dlsym(handle, "ml_train_layer↩ _pluggable"));error_msg=dlerror();auto close_dl=[handle] { dlclose(handle);};NNTR_THROW_IF_CLEA↩ NUP(error_msg !=nullptr||pluggable==nullptr, std::invalid_argument, close_dl)<< func_tag<< "loading symbol failed, reason: "<< error_msg;auto layer=pluggable-> nntrainer::createfunc ()

  base case of iterate_prop, iterate_prop iterates the given tuple

9.1.1 Detailed Description

This file contains app context related functions and classes that manages the global configuration of the current environment.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

10 November 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

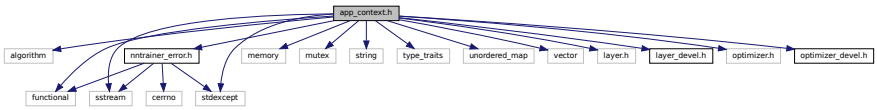
9.2 app_context.h File Reference

This file contains app context related functions and classes that manages the global configuration of the current environment.

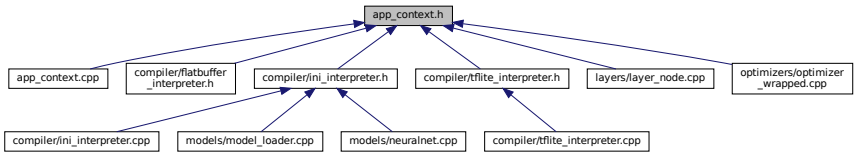
```
#include <algorithm>
#include <functional>
#include <memory>
#include <mutex>
#include <sstream>
#include <stdexcept>
#include <string>
#include <type_traits>
#include <unordered_map>
#include <vector>
```

```
#include <layer.h>
#include <layer_devel.h>
#include <optimizer.h>
#include <optimizer_devel.h>
#include <nntrainer_error.h>
```

Include dependency graph for app_context.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::AppContext

  App.

Namespaces

- nntrainer

Functions

- const template int nntrainer::AppContext::registerFactory< nntrainer::Optimizer > (const FactoryType< nntrainer::Optimizer > factory, const std::string &key, const int int_key)

  int AppContext::registerFactory

- const template int nntrainer::AppContext::registerFactory< nntrainer::Layer > (const FactoryType< nntrainer::Layer > factory, const std::string &key, const int int_key)

  int AppContext::registerFactory

- const template int nntrainer::AppContext::registerFactory< ml::train::LearningRateScheduler > (const FactoryType< ml::train::LearningRateScheduler > factory, const std::string &key, const int int_key)

  int AppContext::registerFactory

9.2.1 Detailed Description

This file contains app context related functions and classes that manages the global configuration of the current environment.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

10 November 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

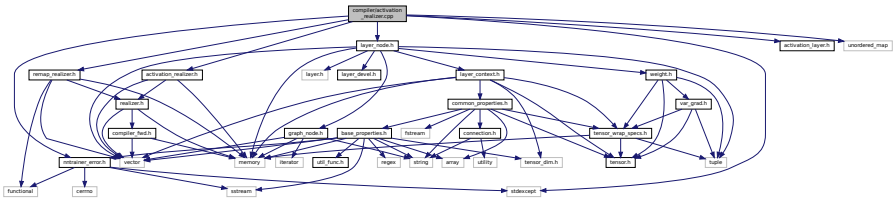
Bug No known bugs except for NYI items

9.3 compiler/activation_realizer.cpp File Reference

NNTrainer graph realizer which realizes activation!=none to actual activation node.

```
#include "nntrainer_error.h"
#include <activation_realizer.h>
#include <remap_realizer.h>
#include <activation_layer.h>
#include <layer_node.h>
#include <stdexcept>
#include <unordered_map>
```

Include dependency graph for activation_realizer.cpp:



Namespaces

- nntrainer

9.3.1 Detailed Description

NNTrainer graph realizer which realizes activation!=none to actual activation node.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

23 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

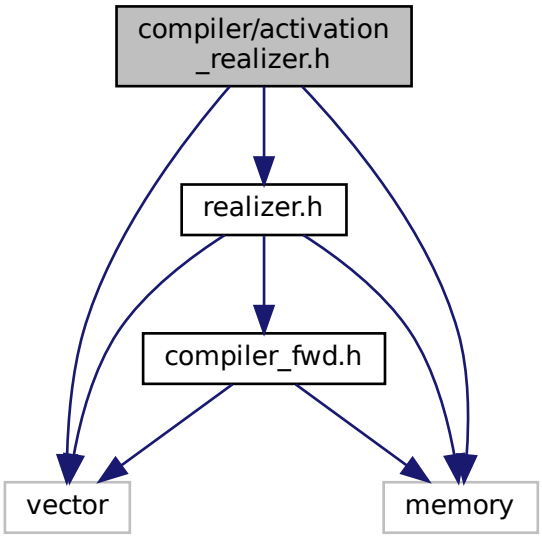
Bug No known bugs except for NYI items

9.4 compiler/activation_realizer.h File Reference

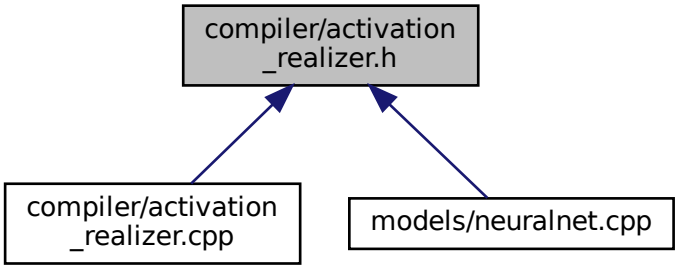
NNTrainer graph realizer which realizes activation!=none to actual activation node.

```
#include <memory>
#include <vector>
#include <realizer.h>
```

Include dependency graph for activation_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::ActivationRealizer

  Graph realizer which realizes activation.

Namespaces

- nntrainer

9.4.1 Detailed Description

NNTrainer graph realizer which realizes activation!=none to actual activation node.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

23 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

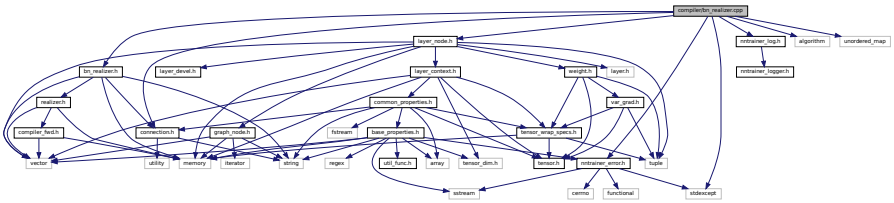
Bug No known bugs except for NYI items

9.5 compiler/bn_realizer.cpp File Reference

NNTrainer graph realizer which remove batch normalization layer for inference.

```
#include <bn_realizer.h>
#include <connection.h>
#include <layer_node.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <algorithm>
#include <stdexcept>
#include <unordered_map>
```

Include dependency graph for bn_realizer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.5.1 Detailed Description

NNTrainer graph realizer which remove batch normalization layer for inference.

Copyright (C) 2022 Jijoong Moon jijoong.moon@samsung.com

Date

13 April 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

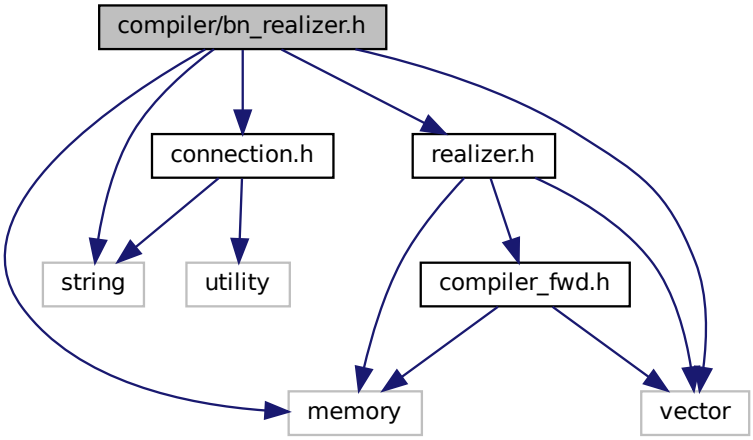
Bug No known bugs except for NYI items

9.6 compiler/bn_realizer.h File Reference

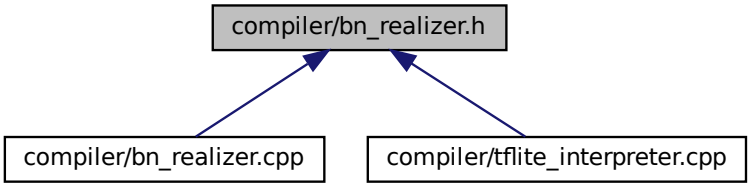
NNTrainer graph realizer which remove batch normalization layer for inference.

```
#include <memory>
#include <string>
#include <vector>
#include <connection.h>
#include <realizer.h>
```

Include dependency graph for bn_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::BnRealizer

  Graph realizer class which removes batch normalization layer from the graph.

Namespaces

- nntrainer

9.6.1 Detailed Description

NNTrainer graph realizer which remove batch normalization layer for inference.

Copyright (C) 2022 Jijoong Moon jijoong.moon@samsung.com

Date

13 April 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

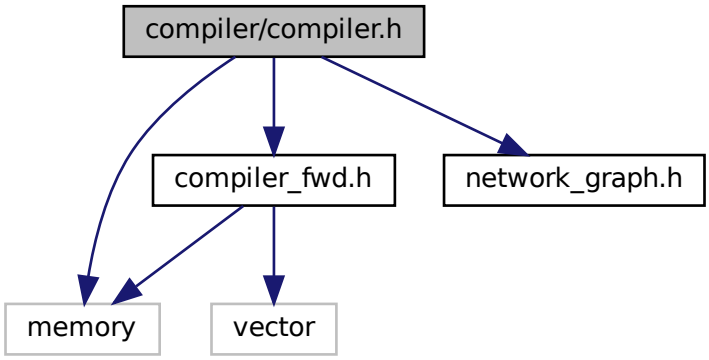
Bug No known bugs except for NYI items

9.7 compiler/compiler.h File Reference

NNTrainer compiler that reads to generate optimized graph.

```
#include <memory>
#include <compiler_fwd.h>
#include <network_graph.h>
```

Include dependency graph for compiler.h:



Classes

- class nntrainer::GraphCompiler

  Pure virtual class for the Graph Compiler.

Namespaces

- nntrainer

9.7.1 Detailed Description

NNTrainer compiler that reads to generate optimized graph.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

01 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

Graph is convertible either to iostream, representation, executable by appropriate compiler and interpreter For example, if istream would be from a a.tflite file,

GraphRepresentation g; GraphCompiler ∗ compiler = new NNTrainerCPUCompiler;

ExecutableGraph eg = compiler->compile(g);

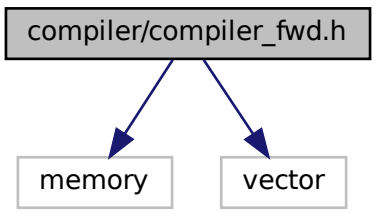
+------+–+-------+ |GraphRepresentation| +------+----------+ | ^ compile() | | | | (Compiler) | | | | decompile() v | +-------+------+ |ExecutableGraph| +-------------+

9.8 compiler/compiler_fwd.h File Reference

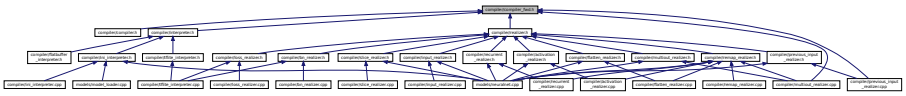
NNTrainer graph compiler related forward declarations.

```
#include <memory>
#include <vector>
```

Include dependency graph for compiler_fwd.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- nntrainer

Typedefs

- using **nntrainer::GraphRepresentation** = std::vector< std::shared_ptr< LayerNode > >
- using **nntrainer::ExecutableGraph** = NetworkGraph

9.8.1 Detailed Description

NNTrainer graph compiler related forward declarations.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

9.9 compiler/flatbuffer_interpreter.cpp File Reference

NNTrainer ∗.flatbuffer Interpreter.

9.9.1 Detailed Description

NNTrainer ∗.flatbuffer Interpreter.

Copyright (C) 2023 DongHak Park donghak.park@samsung.com

Date

09 February 2023

See also

https://github.com/nnstreamer/nntrainer

Author

Donghak Park donghak.park@samsung.com

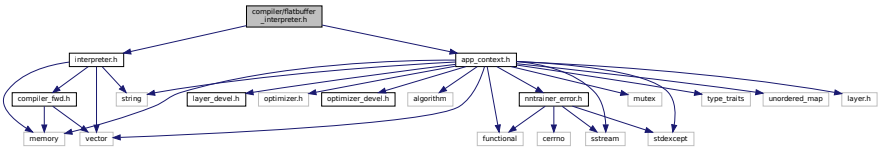
Bug No known bugs except for NYI items

9.10 compiler/flatbuffer_interpreter.h File Reference

NNTrainer flatbuffer Interpreter.

```
#include <app_context.h>
#include <interpreter.h>
```

Include dependency graph for flatbuffer_interpreter.h:



Classes

- class nntrainer::FlatBufferInterpreter

  flatbuffer graph interpreter class

Namespaces

- nntrainer

9.10.1 Detailed Description

NNTrainer flatbuffer Interpreter.

Copyright (C) 2023 DongHak Park donghak.park@samsung.com

Date

09 February 2023

See also

https://github.com/nnstreamer/nntrainer

Author

Donghak Park donghak.park@samsung.com

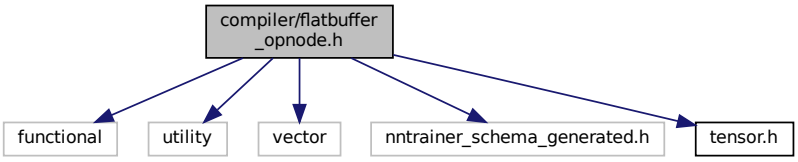
Bug No known bugs except for NYI items

9.11 compiler/flatbuffer_opnode.h File Reference

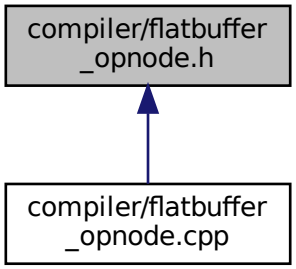
NNTrainer flatbuffer opnode.

```
#include <functional>
#include <utility>
#include <vector>
#include <nntrainer_schema_generated.h>
#include <tensor.h>
```

Include dependency graph for flatbuffer_opnode.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::FlatBufferOpNode

  FlatBufferOpNode class.

Namespaces

- nntrainer

9.11.1 Detailed Description

NNTrainer flatbuffer opnode.

Copyright (C) 2023 DongHak Park donghak.park@samsung.com

Date

10 February 2023

See also

https://github.com/nnstreamer/nntrainer

Author

Donghak Park donghak.park@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2023 DongHak Park donghak.park@samsung.com

Date

10 February 2023

See also

https://github.com/nnstreamer/nntrainer

Author

Donghak Park donghak.park@samsung.com

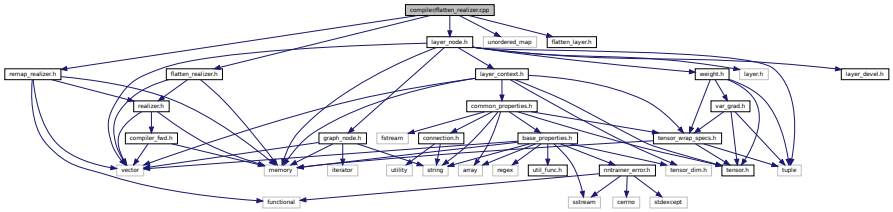
Bug No known bugs except for NYI items

9.12 compiler/flatten_realizer.cpp File Reference

NNTrainer graph realizer which realizes flatten=true to actual node.

```
#include <flatten_realizer.h>
#include <remap_realizer.h>
#include <unordered_map>
#include <flatten_layer.h>
#include <layer_node.h>
```

Include dependency graph for flatten_realizer.cpp:



Namespaces

- nntrainer

9.12.1 Detailed Description

NNTrainer graph realizer which realizes flatten=true to actual node.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

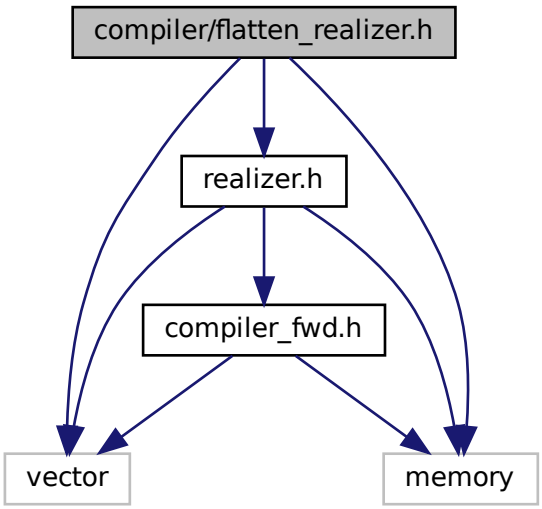
Bug No known bugs except for NYI items

9.13 compiler/flatten_realizer.h File Reference

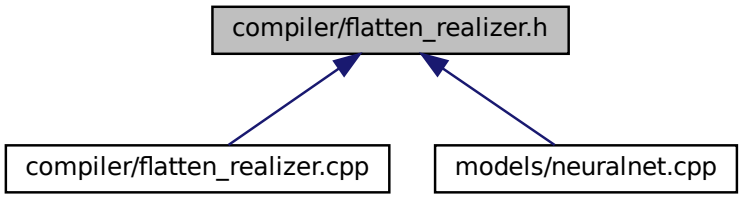
NNTrainer graph realizer which realizes flatten=true to actual node.

```
#include <memory>
#include <vector>
#include <realizer.h>
```

Include dependency graph for flatten_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::FlattenRealizer

  Graph realizer class.

Namespaces

- nntrainer

9.13.1 Detailed Description

NNTrainer graph realizer which realizes flatten=true to actual node.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

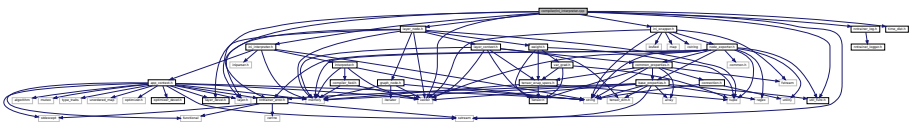
Bug No known bugs except for NYI items

9.14 compiler/ini_interpreter.cpp File Reference

NNTrainer Ini Interpreter (partly moved from model_loader.c)

```
#include <ini_interpreter.h>
#include <sstream>
#include <vector>
#include <ini_wrapper.h>
#include <layer.h>
#include <layer_node.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <time_dist.h>
#include <util_func.h>
```

Include dependency graph for ini_interpreter.cpp:



Namespaces

- nntrainer

Variables

- static constexpr const char * **FUNC_TAG** = "[IniInterpreter] "
- static constexpr const char * **UNKNOWN_STR** = "UNKNOWN"
- static constexpr const char * **NONE_STR** = "NONE"
- static constexpr const char * **MODEL_STR** = "model"
- static constexpr const char * **DATASET_STR** = "dataset"
- static constexpr const char * **TRAINSET_STR** = "train_set"
- static constexpr const char * **VALIDSET_STR** = "valid_set"
- static constexpr const char * **TESTSET_STR** = "test_set"
- static constexpr const char * **OPTIMIZER_STR** = "optimizer"
- static constexpr const char * **LRSCHED_STR** = "LearningRateScheduler"

9.14.1 Detailed Description

NNTrainer Ini Interpreter (partly moved from model_loader.c)

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

02 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com
Parichay Kapoor pk.kapoor@samsung.com
Jihoon Lee jhoon.it.lee@samsung.com

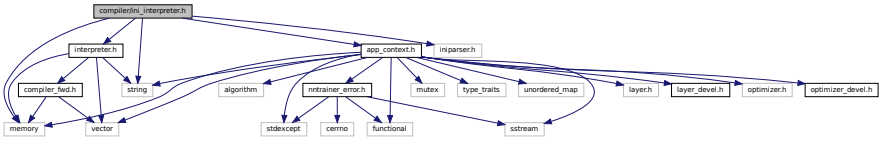
Bug No known bugs except for NYI items

9.15 compiler/ini_interpreter.h File Reference

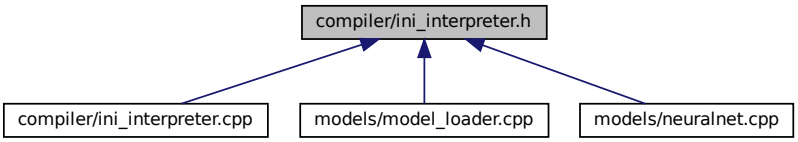
NNTrainer Ini Interpreter.

```
#include <memory>
#include <string>
#include <iniparser.h>
#include <app_context.h>
#include <interpreter.h>
```

Include dependency graph for ini_interpreter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::IniGraphInterpreter

  ini graph interpreter class

Namespaces

- nntrainer

9.15.1 Detailed Description

NNTrainer Ini Interpreter.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

02 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

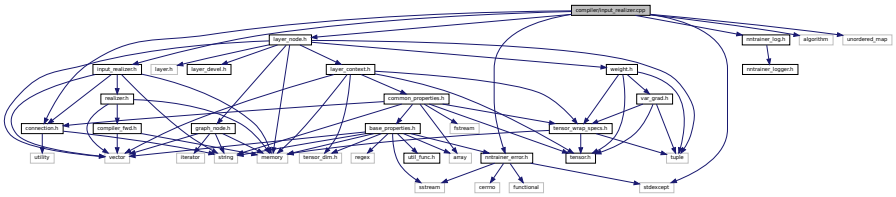
Bug No known bugs except for NYI items

9.16 compiler/input_realizer.cpp File Reference

NNTrainer graph realizer which remaps input to the external graph.

```
#include <connection.h>
#include <input_realizer.h>
#include <layer_node.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <algorithm>
#include <stdexcept>
#include <unordered_map>
```

Include dependency graph for input_realizer.cpp:



Namespaces

- nntrainer

9.16.1 Detailed Description

NNTrainer graph realizer which remaps input to the external graph.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

14 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

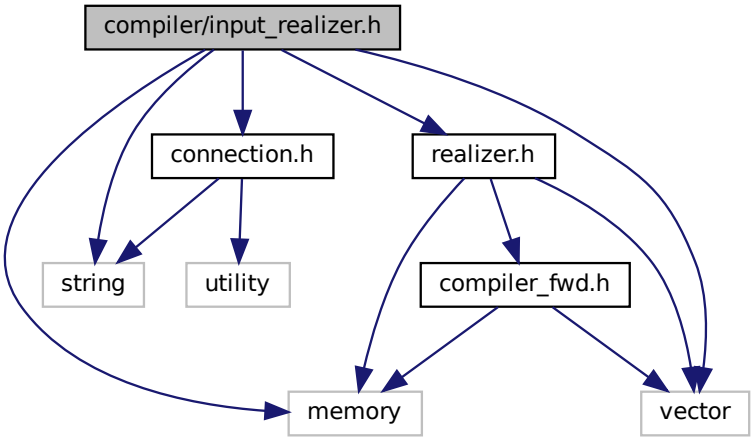
Bug No known bugs except for NYI items

9.17 compiler/input_realizer.h File Reference

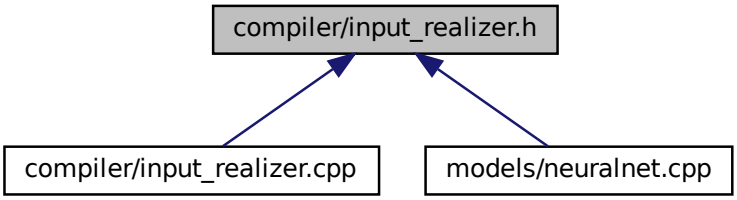
NNTrainer graph realizer which remaps input to the external graph.

```
#include <memory>
#include <string>
#include <vector>
#include <connection.h>
#include <realizer.h>
```

Include dependency graph for input_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::InputRealizer

  Graph realizer class which remaps input from start -> input layers.

Namespaces

- nntrainer

9.17.1 Detailed Description

NNTrainer graph realizer which remaps input to the external graph.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

14 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

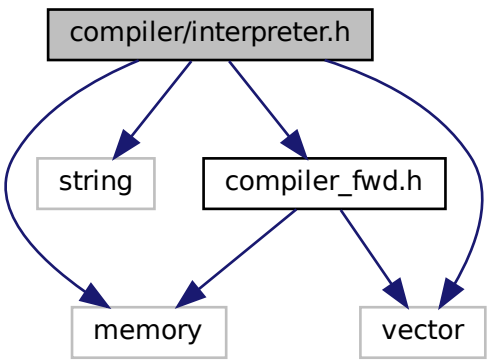
Bug No known bugs except for NYI items

9.18 compiler/interpreter.h File Reference

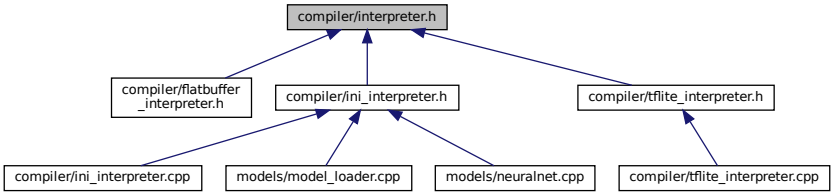
NNTrainer interpreter that reads and generates a graphRepresentation from a file.

```
#include <memory>
#include <string>
#include <vector>
#include <compiler_fwd.h>
```

Include dependency graph for interpreter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::GraphInterpreter

  Pure virtual class for the Graph Interpreter.

Namespaces

- nntrainer

9.18.1 Detailed Description

NNTrainer interpreter that reads and generates a graphRepresentation from a file.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

01 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

Note

The boundary of graph interpreter is restricted to graph only.

Graph is convertible either from a file, representation by a appropriate interpreter For example, if istream would be from a a.tflite file,

GraphRepresentation g; GraphInterpreter * interpreter = new TfliteInterpreter;

std::ifstream f = std::open("a.tflite"); g = interpreter->serialize(f);

```
+--------+
|iostream|
+--+--+--+
   ^  |
```

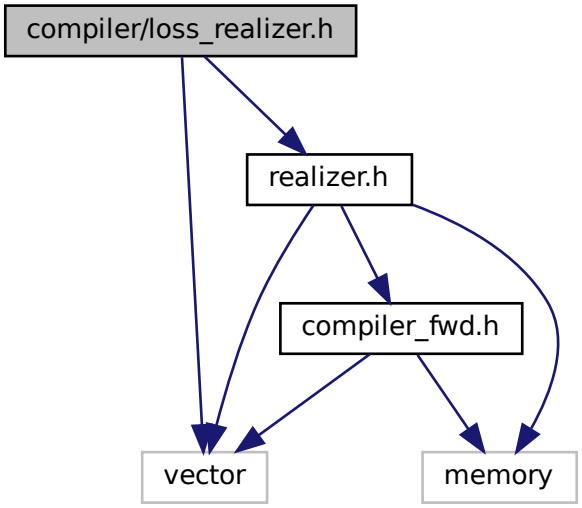
serialize()| | | | (Interpreter) | | | | deserialize() | v +------—+–+-------—+ |GraphRepresentation| +------—+----------—+

9.19 compiler/loss_realizer.h File Reference

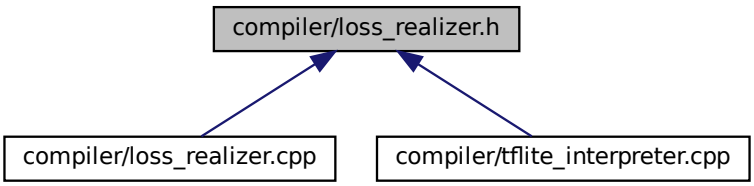
NNTrainer graph realizer which remove loss layer for inference.

```
#include <vector>
#include <realizer.h>
```

Include dependency graph for loss_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::LossRealizer

  Graph realizer class which removes loss layer from the graph.

Namespaces

- nntrainer

9.19.1 Detailed Description

NNTrainer graph realizer which remove loss layer for inference.

Copyright (C) 2022 seongwoo mhs4670go@naver.com

Date

4 May 2022

See also

https://github.com/nnstreamer/nntrainer

Author

seongwoo mhs4670go@naver.com

Bug No known bugs except for NYI items

Copyright (C) 2022 seongwoo mhs4670go@naver.com

Date

4 May 2022

See also

https://github.com/nnstreamer/nntrainer

Author

seongwoo mhs4670go@naver.com

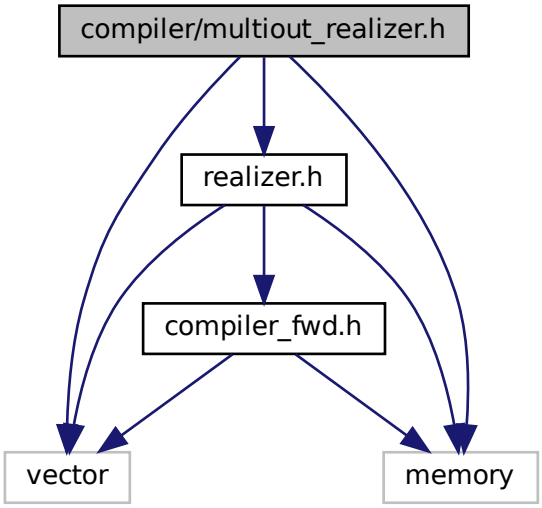
Bug No known bugs except for NYI items

9.20 compiler/multiout_realizer.h File Reference

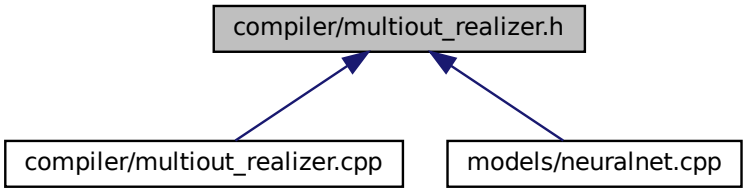
NNTrainer graph realizer which realizes multiout to actual node.

```
#include <memory>
#include <vector>
#include <realizer.h>
```

Include dependency graph for multiout_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::MultioutRealizer

  Add multiout layer when a certain input is referenced multiple times.

Namespaces

- nntrainer

9.20.1 Detailed Description

NNTrainer graph realizer which realizes multiout to actual node.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

17 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

17 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

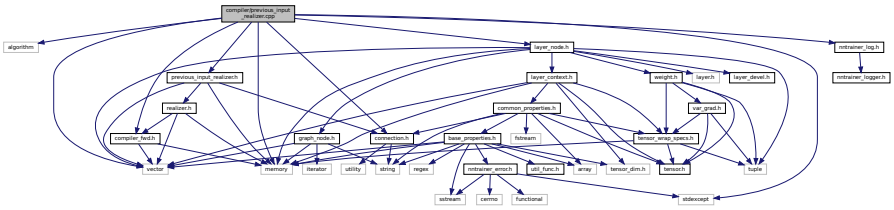
Bug No known bugs except for NYI items

9.21 compiler/previous_input_realizer.cpp File Reference

NNTrainer graph realizer which connects input to previous one if empty.

```
#include <algorithm>
#include <compiler_fwd.h>
#include <memory>
#include <stdexcept>
#include <vector>
#include <connection.h>
#include <layer_node.h>
#include <nntrainer_log.h>
#include <previous_input_realizer.h>
```

Include dependency graph for previous_input_realizer.cpp:



Namespaces

- nntrainer

9.21.1 Detailed Description

NNTrainer graph realizer which connects input to previous one if empty.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

18 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

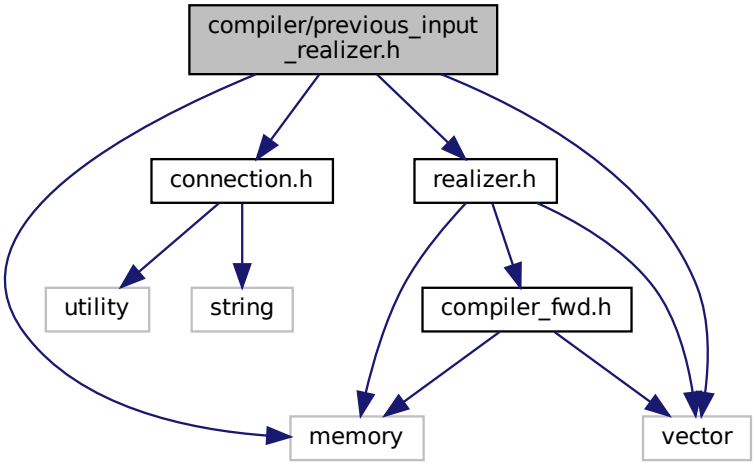
Bug No known bugs except for NYI items

9.22 compiler/previous_input_realizer.h File Reference

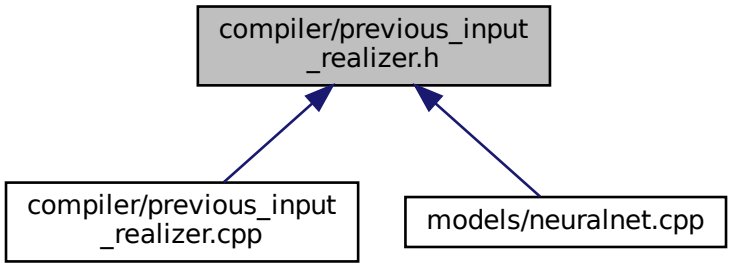
NNTrainer graph realizer which connects input to previous one if empty.

```
#include <memory>
#include <vector>
#include <connection.h>
#include <realizer.h>
```

Include dependency graph for previous_input_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::PreviousInputRealizer

  Add default inputs if input connection is empty. if a node is identified as input with identified_input by user or a node has input_shape property, adding input behavior is skipped.

Namespaces

- nntrainer

9.22.1 Detailed Description

NNTrainer graph realizer which connects input to previous one if empty.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

18 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

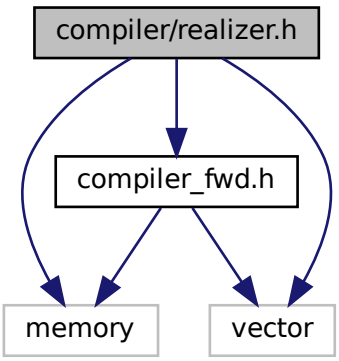
Bug No known bugs except for NYI items

9.23 compiler/realizer.h File Reference

NNTrainer graph realizer which preprocess graph representation as a lowering process of compile.

```
#include <memory>
#include <vector>
#include <compiler_fwd.h>
```

Include dependency graph for realizer.h:



This graph shows which files directly or indirectly include this file:

Classes

- class nntrainer::GraphRealizer

  Graph realizer class.

Namespaces

- nntrainer

9.23.1 Detailed Description

NNTrainer graph realizer which preprocess graph representation as a lowering process of compile.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

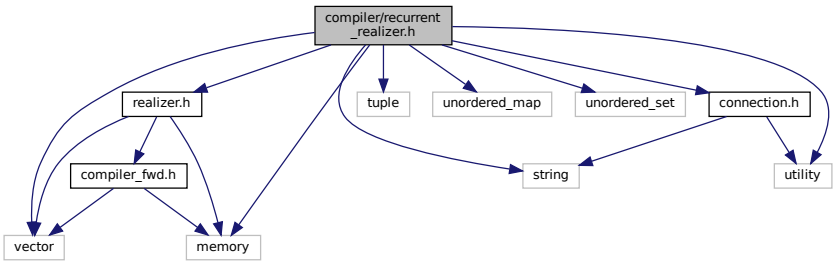
9.24 compiler/recurrent_realizer.h File Reference

NNTrainer graph realizer to create unrolled graph from a graph realizer.

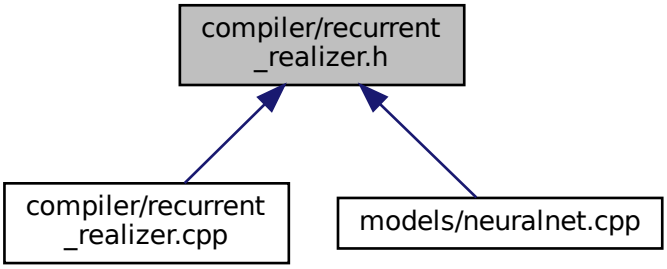
```
#include <realizer.h>
#include <memory>
#include <string>
#include <tuple>
#include <unordered_map>
#include <unordered_set>
#include <utility>
#include <vector>
```

```
#include <connection.h>
```

Include dependency graph for recurrent_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::RecurrentRealizer

  Recurrent Realizer which unrolls graph from given graph representation.

Namespaces

- nntrainer

9.24.1 Detailed Description

NNTrainer graph realizer to create unrolled graph from a graph realizer.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

12 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

Date

12 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

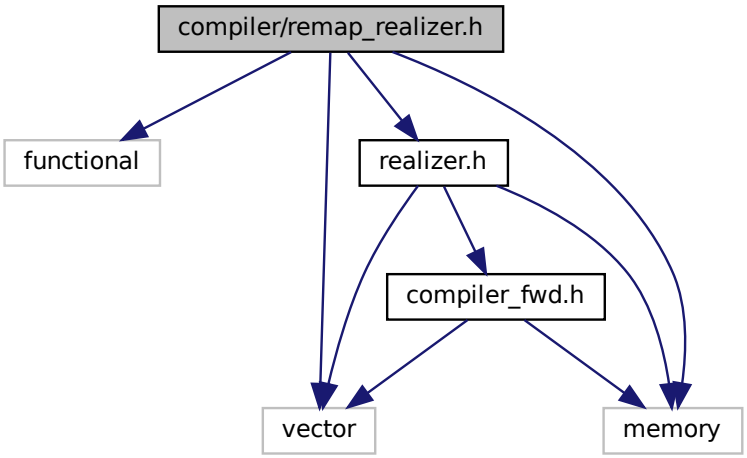
Bug No known bugs except for NYI items

9.25 compiler/remap_realizer.h File Reference

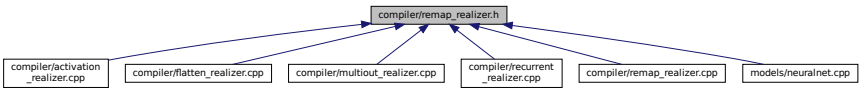
NNTrainer graph realizer which realizes identifier to a new identifier.

```
#include <functional>
#include <memory>
#include <vector>
#include <realizer.h>
```

Include dependency graph for remap_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::RemapRealizer

  Graph realizer class which remaps identifiers inside the graph representation, remap_function will be applied for all the layers identifier visible.

Namespaces

- nntrainer

9.25.1 Detailed Description

NNTrainer graph realizer which realizes identifier to a new identifier.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

12 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

12 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

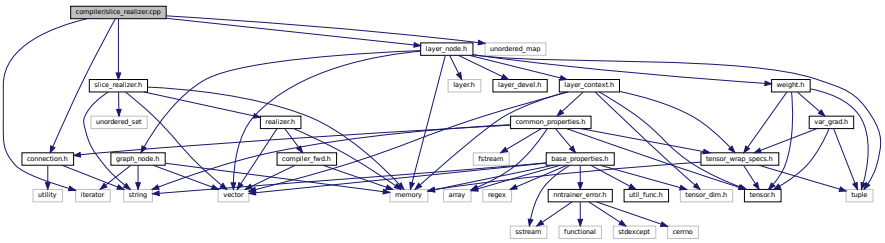
Bug No known bugs except for NYI items

9.26 compiler/slice_realizer.cpp File Reference

NNTrainer graph realizer which slice the graph representation.

```
#include <connection.h>
#include <iterator>
#include <layer_node.h>
#include <slice_realizer.h>
#include <unordered_map>
```

Include dependency graph for slice_realizer.cpp:



Namespaces

- nntrainer

9.26.1 Detailed Description

NNTrainer graph realizer which slice the graph representation.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

14 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

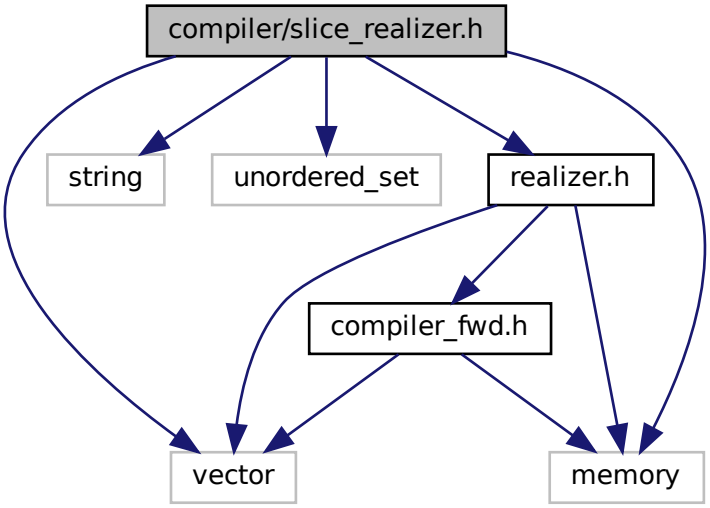
Bug No known bugs except for NYI items

9.27 compiler/slice_realizer.h File Reference

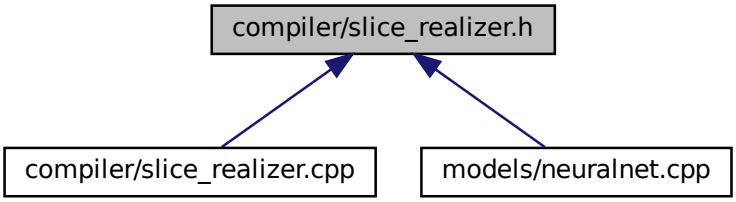
NNTrainer graph realizer which slice the graph representation.

```
#include <memory>
#include <string>
#include <unordered_set>
#include <vector>
#include <realizer.h>
```

Include dependency graph for slice_realizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::SliceRealizer

  Graph realizer class which slice graph representation.

Namespaces

- nntrainer

9.27.1 Detailed Description

NNTrainer graph realizer which slice the graph representation.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

14 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

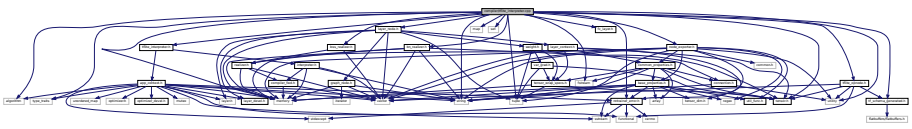
Bug No known bugs except for NYI items

9.28 compiler/tflite_interpreter.cpp File Reference

NNTrainer ∗.tflite Interpreter.

```
#include <tflite_interpreter.h>
#include <algorithm>
#include <fstream>
#include <map>
#include <memory>
#include <set>
#include <string>
#include <tuple>
#include <type_traits>
#include <utility>
#include <bn_realizer.h>
#include <fc_layer.h>
#include <layer_node.h>
#include <loss_realizer.h>
#include <nntrainer_error.h>
#include <node_exporter.h>
#include <tensor.h>
#include <tf_schema_generated.h>
#include <tflite_opnode.h>
```

Include dependency graph for tflite_interpreter.cpp:



Namespaces

- nntrainer

Variables

- static constexpr const char * **FUNC_TAG** = "[TFLITE INTERPRETER] "

9.28.1 Detailed Description

NNTrainer *.tflite Interpreter.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

12 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

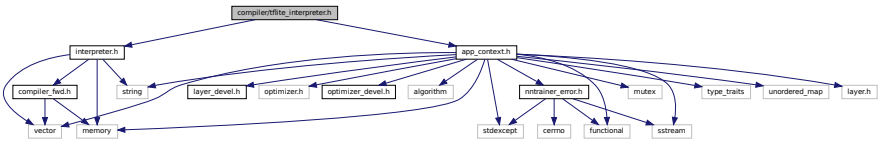
Bug No known bugs except for NYI items

9.29 compiler/tflite_interpreter.h File Reference

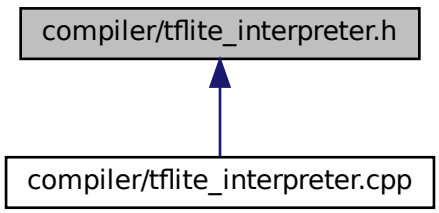
NNTrainer *.tflite Interpreter.

```
#include <interpreter.h>
#include <app_context.h>
```

Include dependency graph for tflite_interpreter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::TfliteInterpreter

  tflite graph interpreter class

Namespaces

- nntrainer

9.29.1 Detailed Description

NNTrainer ∗.tflite Interpreter.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

12 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

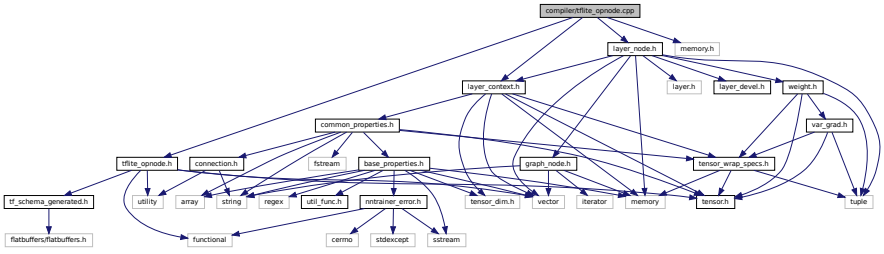
Bug No known bugs except for NYI items

9.30 compiler/tflite_opnode.cpp File Reference

contains tflite opnode which has information to convert to tflite file

```
#include <tflite_opnode.h>
#include <layer_context.h>
#include <layer_node.h>
#include <memory.h>
```

Include dependency graph for tflite_opnode.cpp:



Namespaces

- nntrainer

9.30.1 Detailed Description

contains tflite opnode which has information to convert to tflite file

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

28 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Donghak Park donghak.park@samsung.com

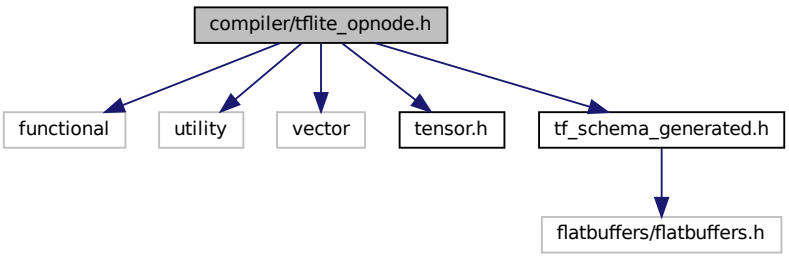
Bug No known bugs except for NYI items

9.31 compiler/tflite_opnode.h File Reference

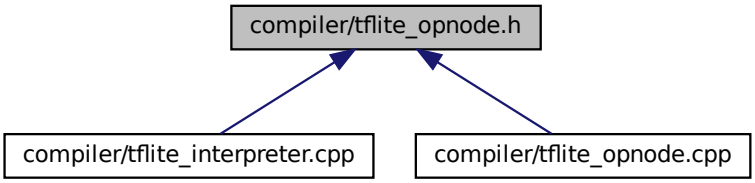
contains tflite opnode which has information to convert to tflite file

```
#include <functional>
#include <utility>
#include <vector>
#include <tensor.h>
#include <tf_schema_generated.h>
```

Include dependency graph for tflite_opnode.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::TfOpNode

  tensorflow operational node representation. This class contains, information to build operation flatbuffer

Namespaces

- nntrainer

9.31.1 Detailed Description

contains tflite opnode which has information to convert to tflite file

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

28 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Donghak Park donghak.park@samsung.com

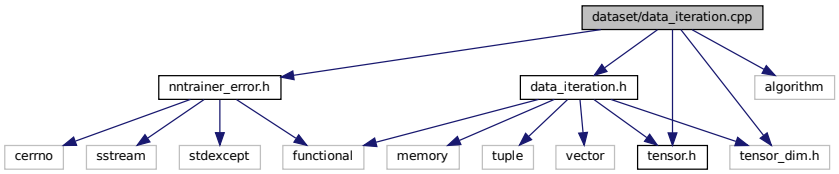
Bug No known bugs except for NYI items

9.32 dataset/data_iteration.cpp File Reference

This file contains iteration and sample class.

```
#include <data_iteration.h>
#include <algorithm>
#include <nntrainer_error.h>
#include <tensor.h>
#include <tensor_dim.h>
```

Include dependency graph for data_iteration.cpp:



Namespaces

- nntrainer

9.32.1 Detailed Description

This file contains iteration and sample class.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

11 Aug 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

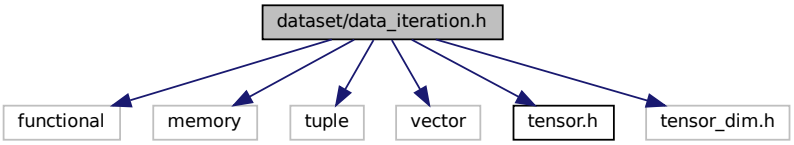
Bug No known bugs except for NYI items

9.33 dataset/data_iteration.h File Reference

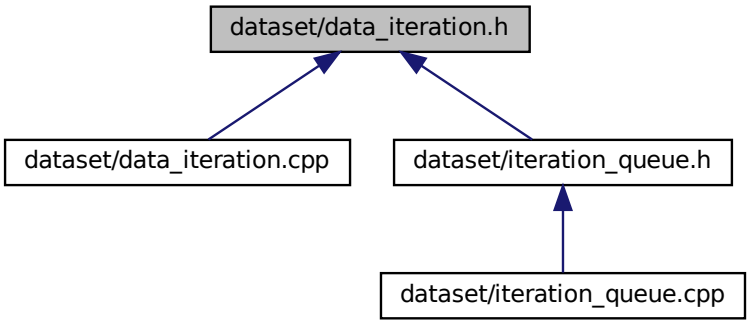
This file contains iteration and sample class.

```
#include <functional>
#include <memory>
#include <tuple>
#include <vector>
#include <tensor.h>
#include <tensor_dim.h>
```

Include dependency graph for data_iteration.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::Iteration

  Iteration class which owns the memory chunk for a single batch.
- class nntrainer::Sample

  Sample class which views the memory for a single sample.

Namespaces

- nntrainer

9.33.1 Detailed Description

This file contains iteration and sample class.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

11 Aug 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

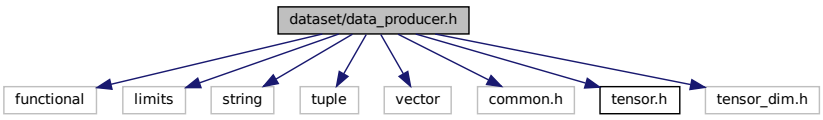
Bug No known bugs except for NYI items

9.34 dataset/data_producer.h File Reference

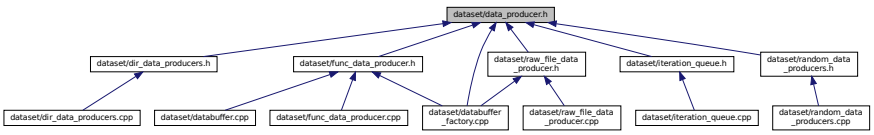
This file contains data producer interface.

```
#include <functional>
#include <limits>
#include <string>
#include <tuple>
#include <vector>
#include <common.h>
#include <tensor.h>
#include <tensor_dim.h>
```

Include dependency graph for data_producer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::DataProducer

  DataProducer interface used to abstract data provider.

Namespaces

- nntrainer

9.34.1 Detailed Description

This file contains data producer interface.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

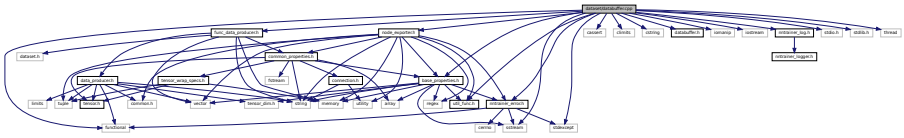
Bug No known bugs except for NYI items

9.35 dataset/databuffer.cpp File Reference

This is buffer object to handle big data.

```
#include <base_properties.h>
#include <cassert>
#include <climits>
#include <cstring>
#include <databuffer.h>
#include <func_data_producer.h>
#include <functional>
#include <iomanip>
#include <iostream>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <sstream>
#include <stdexcept>
#include <stdio.h>
#include <stdlib.h>
#include <thread>
#include <util_func.h>
```

Include dependency graph for databuffer.cpp:



Classes

- class nntrainer::PropsBufferSize

  Props containing buffer size value.

Namespaces

- nntrainer

Variables

- constexpr char **nntrainer::USER_DATA** [ ] = "user_data"

9.35.1 Detailed Description

This is buffer object to handle big data.

Date

04 December 2019

See also

https://github.com/nnstreamer/nntrainer

Author

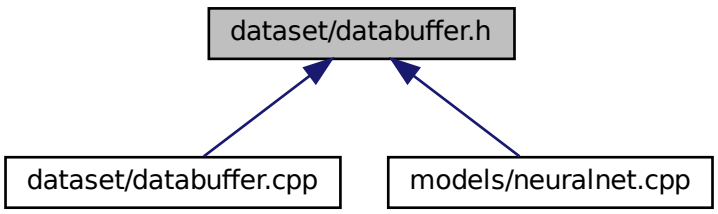
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.36 dataset/databuffer.h File Reference

This is buffer object to handle big data.

This graph shows which files directly or indirectly include this file:



9.36.1 Detailed Description

This is buffer object to handle big data.

Date

04 December 2019

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

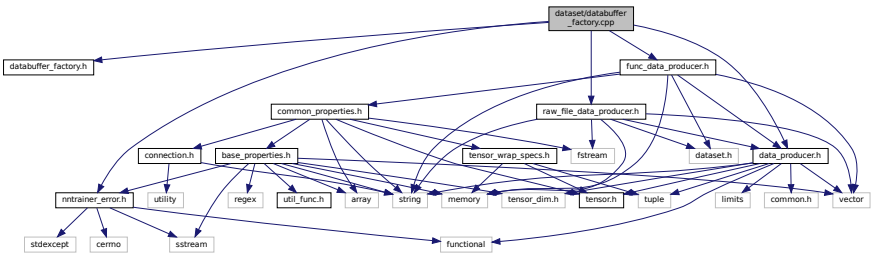
Bug No known bugs except for NYI items

9.37 dataset/databuffer_factory.cpp File Reference

This is the databuffer factory.

```
#include <databuffer_factory.h>
#include <data_producer.h>
#include <func_data_producer.h>
#include <nntrainer_error.h>
#include <raw_file_data_producer.h>
```

Include dependency graph for databuffer_factory.cpp:



Namespaces

- nntrainer

Functions

- std::unique_ptr< DataBuffer > nntrainer::createDataBuffer (DatasetType type)

  Factory creator with constructor.
- std::unique_ptr< DataBuffer > nntrainer::createDataBuffer (DatasetType type, const char *file)

  Factory creator with constructor for dataset.
- std::unique_ptr< DataBuffer > nntrainer::createDataBuffer (DatasetType type, datagen_cb cb, void *user_data)

  Factory creator with constructor for dataset.

9.37.1 Detailed Description

This is the databuffer factory.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

11 October 2020

See also

https://github.com/nnstreamer/nntrainer

Author

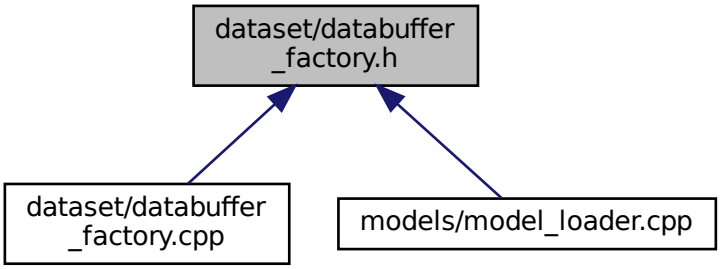
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.38 dataset/databuffer_factory.h File Reference

This is the layer factory.

This graph shows which files directly or indirectly include this file:



9.38.1 Detailed Description

This is the layer factory.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

19 October 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

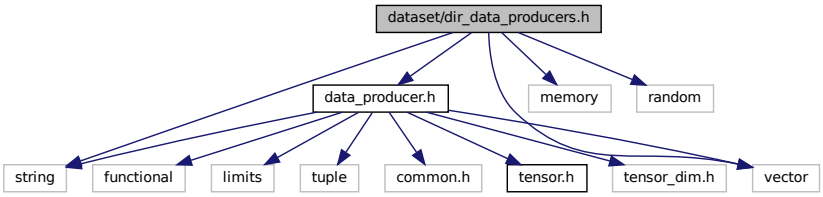
Bug No known bugs except for NYI items

9.39 dataset/dir_data_producers.h File Reference

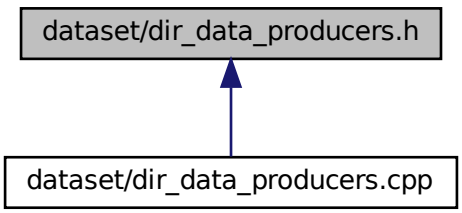
This file contains dir data producers, reading from the files in directory.

```
#include <data_producer.h>
#include <memory>
#include <random>
#include <string>
#include <vector>
```

Include dependency graph for dir_data_producers.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::DirDataProducer

  DirDataProducer which generates a onehot vector as a label.

Namespaces

- nntrainer

9.39.1 Detailed Description

This file contains dir data producers, reading from the files in directory.

Copyright (C) 2023 Jijoong Moon jijoong.moon@samsung.com

Date

24 Feb 2023

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2023 Jijoong Moon jijoong.moon@samsung.com

Date

24 Feb 2023

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

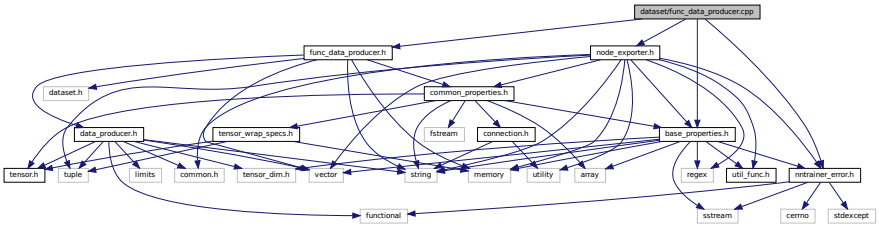
Bug No known bugs except for NYI items

9.40 dataset/func_data_producer.cpp File Reference

This file contains various data producers from a callback.

```
#include <func_data_producer.h>
#include <base_properties.h>
#include <nntrainer_error.h>
#include <node_exporter.h>
```

Include dependency graph for func_data_producer.cpp:



Namespaces

- nntrainer

9.40.1 Detailed Description

This file contains various data producers from a callback.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

12 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

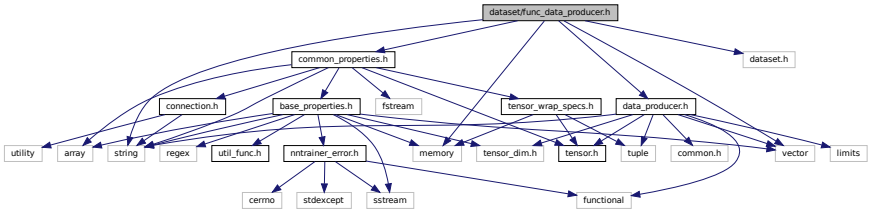
Bug No known bugs except for NYI items

9.41 dataset/func_data_producer.h File Reference

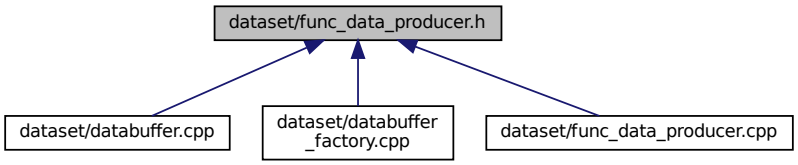
This file contains various data producers from a callback.

```
#include <common_properties.h>
#include <data_producer.h>
#include <dataset.h>
#include <memory>
#include <string>
#include <vector>
```

Include dependency graph for func_data_producer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::FuncDataProducer

  FuncDataProducer which contains a callback and returns back.

Namespaces

- nntrainer

Typedefs

- using **nntrainer::datagen_cb** = ml::train::datagen_cb

9.41.1 Detailed Description

This file contains various data producers from a callback.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

12 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

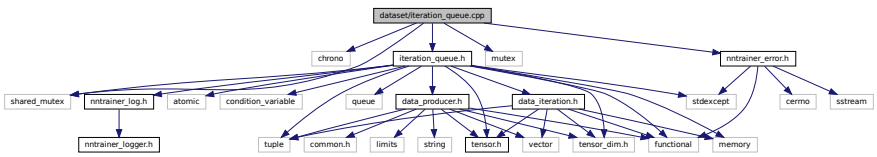
Bug No known bugs except for NYI items

9.42 dataset/iteration_queue.cpp File Reference

This file contains thread safe queue.

```
#include <chrono>
#include <iteration_queue.h>
#include <mutex>
#include <nntrainer_error.h>
#include <shared_mutex>
```

Include dependency graph for iteration_queue.cpp:



Namespaces

- nntrainer

9.42.1 Detailed Description

This file contains thread safe queue.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

13 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

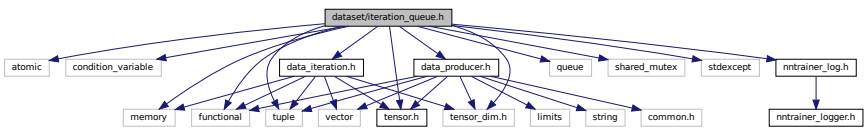
Bug No known bugs except for NYI items

9.43 dataset/iteration_queue.h File Reference

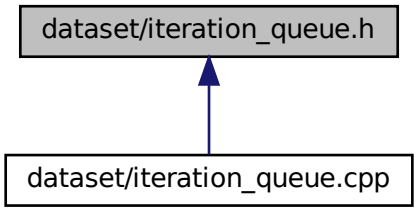
This file contains thread safe queue for a single iteration.

```
#include <atomic>
#include <condition_variable>
#include <functional>
#include <memory>
#include <queue>
#include <shared_mutex>
#include <stdexcept>
#include <tuple>
#include <data_iteration.h>
#include <data_producer.h>
#include <nntrainer_log.h>
#include <tensor.h>
#include <tensor_dim.h>
```

Include dependency graph for iteration_queue.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::ViewQueue< T >

  Thread Safe Queue implementation dedicated for the non-owing pointer.
- class nntrainer::ScopedView< T >

  A view container that calls a callback on destruct.
- class nntrainer::IterationQueue

  Iteration queue that owns the buffer for input / labels.

Namespaces

- nntrainer

9.43.1 Detailed Description

This file contains thread safe queue for a single iteration.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

13 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

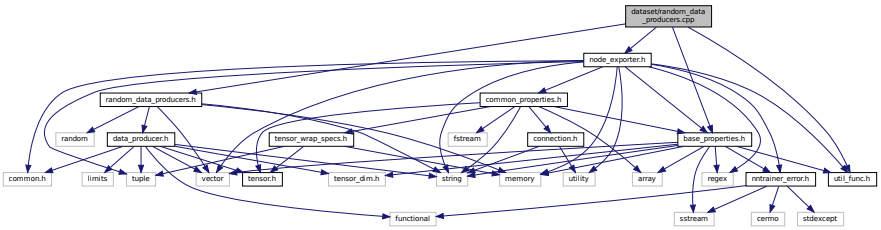
Bug No known bugs except for NYI items

9.44 dataset/random_data_producers.cpp File Reference

This file contains various random data producers.

```
#include <random_data_producers.h>
#include <base_properties.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for random_data_producers.cpp:



Classes

- class nntrainer::PropsMin

  Props containing min value.
- class nntrainer::PropsMax

  Props containing max value.
- class nntrainer::PropsNumSamples

  Props containing number of samples A random data producer has theoretical size. number of samples is used to set theoretical size of the random data producer's data size.

Namespaces

- nntrainer

9.44.1 Detailed Description

This file contains various random data producers.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

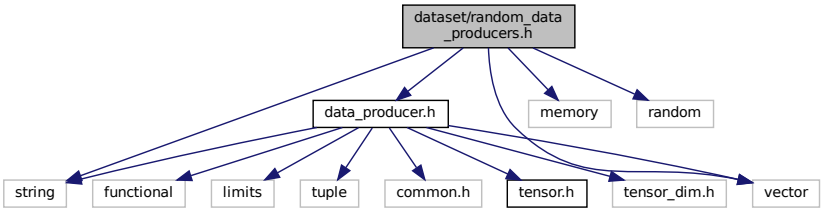
Bug No known bugs except for NYI items

9.45 dataset/random_data_producers.h File Reference

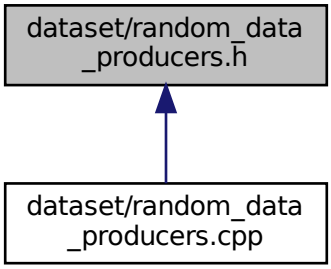
This file contains various random data producers.

```
#include <data_producer.h>
#include <memory>
#include <random>
#include <string>
#include <vector>
```

Include dependency graph for random_data_producers.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::RandomDataOneHotProducer

  RandomDataProducer which generates a onehot vector as a label.

Namespaces

- nntrainer

9.45.1 Detailed Description

This file contains various random data producers.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

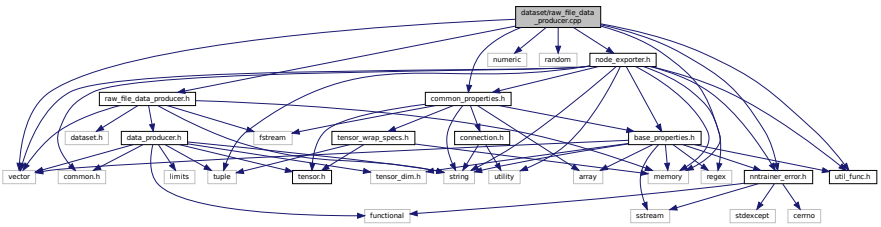
Bug No known bugs except for NYI items

9.46 dataset/raw_file_data_producer.cpp File Reference

This file contains raw file data producers, reading from a file.

```
#include <raw_file_data_producer.h>
#include <memory>
#include <numeric>
#include <random>
#include <vector>
#include <common_properties.h>
#include <nntrainer_error.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for raw_file_data_producer.cpp:



Namespaces

- nntrainer

9.46.1 Detailed Description

This file contains raw file data producers, reading from a file.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

12 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

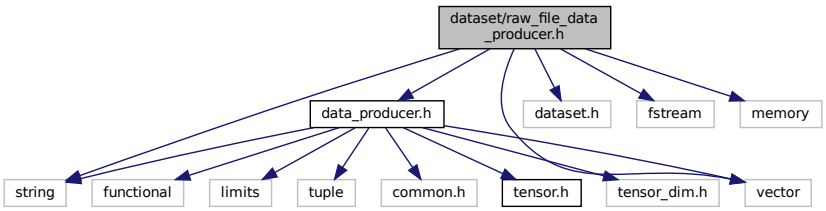
Bug No known bugs except for NYI items

9.47 dataset/raw_file_data_producer.h File Reference

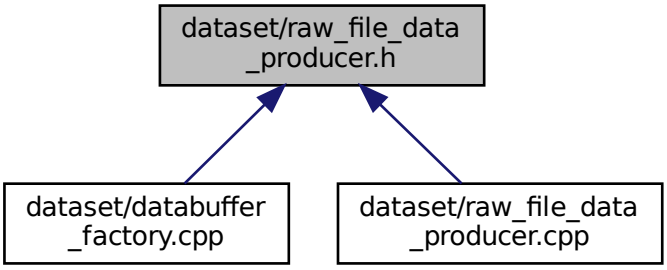
This file contains raw file data producers, reading from a file.

```
#include <data_producer.h>
#include <dataset.h>
#include <fstream>
#include <memory>
#include <string>
#include <vector>
```

Include dependency graph for raw_file_data_producer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::RawFileDataProducer

  RawFileDataProducer which contains a callback and returns back.

Namespaces

- nntrainer

9.47.1 Detailed Description

This file contains raw file data producers, reading from a file.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

12 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

9.48 delegate.h File Reference

This is Delegate Class for the Neural Network.

Classes

- class nntrainer::Backend

  Backend to be used for the operations to use.
- class nntrainer::Device

  Device to be used for the operations to run.
- class nntrainer::DelegateConfig

  Configuration for the delegate.

Namespaces

- nntrainer

9.48.1 Detailed Description

This is Delegate Class for the Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

7 Aug 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Note

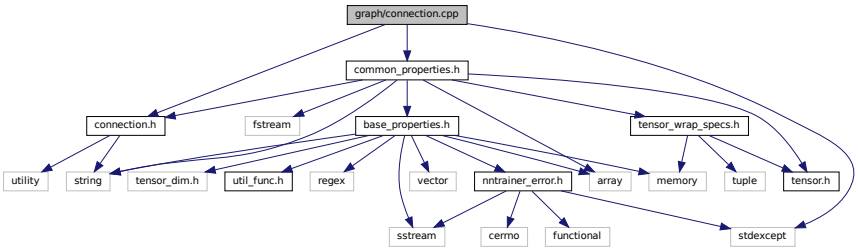
This class is experimental and subject to major modifications.

9.49 graph/connection.cpp File Reference

Connection class and related utility functions.

```
#include <connection.h>
#include <common_properties.h>
#include <stdexcept>
```

Include dependency graph for connection.cpp:



Namespaces

- nntrainer

9.49.1 Detailed Description

Connection class and related utility functions.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

23 Nov 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

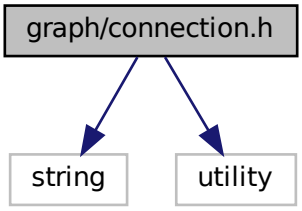
Bug No known bugs except for NYI items

9.50 graph/connection.h File Reference

Connection class and related utility functions.

```
#include <string>
#include <utility>
```

Include dependency graph for connection.h:



This graph shows which files directly or indirectly include this file:

Classes

- class nntrainer::Connection

  RAII class to define a connection.
- struct std::hash< nntrainer::Connection >

  hash specialization for connection

Namespaces

- nntrainer

9.50.1 Detailed Description

Connection class and related utility functions.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

23 Nov 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

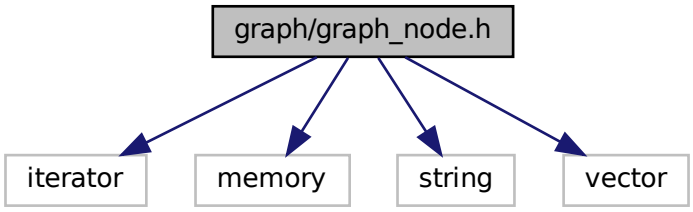
Bug No known bugs except for NYI items

9.51 graph/graph_node.h File Reference

This is the graph node interface for c++ API.

```
#include <iterator>
#include <memory>
#include <string>
#include <vector>
```

Include dependency graph for graph_node.h:



This graph shows which files directly or indirectly include this file:

Classes

- class nntrainer::GraphNode
- class nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >

  Iterator for GraphNode which return const std::shared_ptr<LayerNodeType> object upon realize.
- class nntrainer::GraphNodeReverseIterator< T_iterator >

  Reverse Iterator for GraphNode which return LayerNode object upon realize.

Namespaces

- nntrainer

Typedefs

- template<class LayerNodeType >
  using nntrainer::graph_const_iterator = GraphNodeIterator< LayerNodeType, const std::shared_ptr< GraphNode > >

  Iterators to traverse the graph.
- template<class LayerNodeType >
  using nntrainer::graph_const_reverse_iterator = GraphNodeReverseIterator< GraphNodeIterator< Layer↩ NodeType, const std::shared_ptr< GraphNode > >>

  Iterators to traverse the graph.

9.51.1 Detailed Description

This is the graph node interface for c++ API.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

1 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

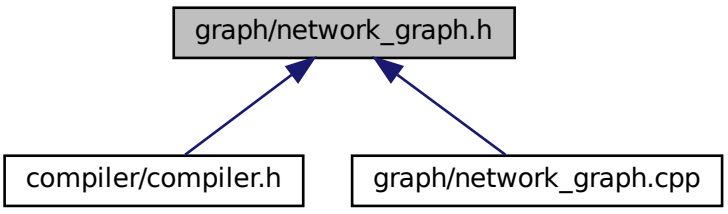
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.52 graph/network_graph.h File Reference

This is Graph Core Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.52.1 Detailed Description

This is Graph Core Class for Neural Network.

This is Network Graph Class for Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

12 May 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

12 May 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

19 Oct 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

Todo Support multi-input graph.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

19 Oct 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

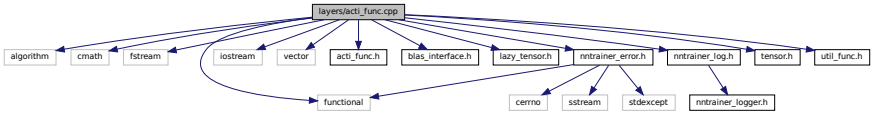
Bug No known bugs except for NYI items

9.53 layers/acti_func.cpp File Reference

This is Activation Layer Class for Neural Network.

```
#include <algorithm>
#include <cmath>
#include <fstream>
#include <functional>
#include <iostream>
#include <vector>
#include <acti_func.h>
#include <blas_interface.h>
#include <lazy_tensor.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <tensor.h>
#include <util_func.h>
```

Include dependency graph for acti_func.cpp:



Namespaces

- nntrainer

Variables

- constexpr static float **nntrainer::NEGATIVE_SLOPE** = 0.01f

9.53.1 Detailed Description

This is Activation Layer Class for Neural Network.

This is Activation Function Class for Neural Network.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

22 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

Date

22 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com
Jijoong Moon jijoong.moon@samsung.com

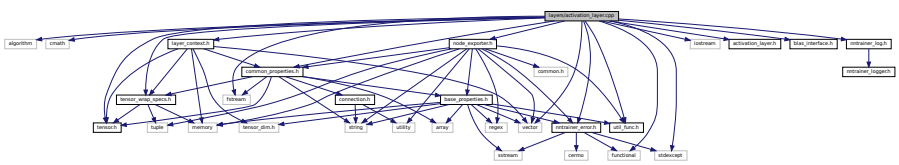
Bug No known bugs except for NYI items

9.54 layers/activation_layer.cpp File Reference

This is Activation Layer Class for Neural Network.

```
#include <algorithm>
#include <cmath>
#include <fstream>
#include <functional>
#include <iostream>
#include <stdexcept>
#include <vector>
#include <activation_layer.h>
#include <blas_interface.h>
#include <common_properties.h>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <tensor.h>
#include <tensor_wrap_specs.h>
#include <util_func.h>
```

Include dependency graph for activation_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.54.1 Detailed Description

This is Activation Layer Class for Neural Network.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

17 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

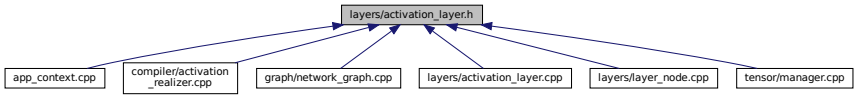
Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

9.55 layers/activation_layer.h File Reference

This is Activation Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.55.1 Detailed Description

This is Activation Layer Class for Neural Network.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

17 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

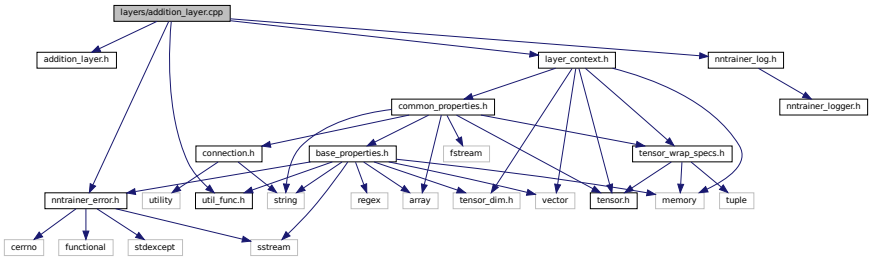
Bug No known bugs except for NYI items

9.56 layers/addition_layer.cpp File Reference

This is Addition Layer Class for Neural Network.

```
#include <addition_layer.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <util_func.h>
#include <layer_context.h>
```

Include dependency graph for addition_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.56.1 Detailed Description

This is Addition Layer Class for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

30 July 2020

See also

https://github.com/nnstreamer/nntrainer

Author

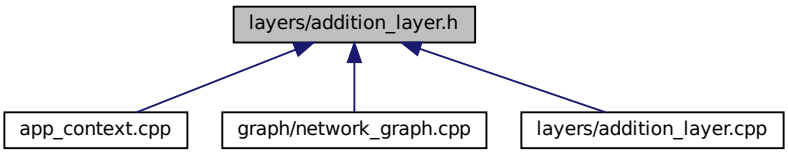
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.57 layers/addition_layer.h File Reference

This is Addition Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.57.1 Detailed Description

This is Addition Layer Class for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

30 July 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

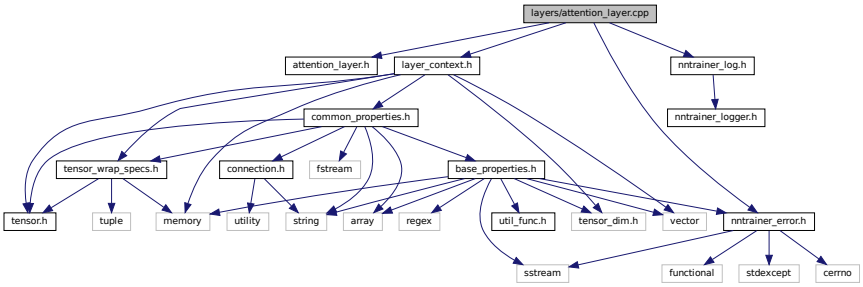
Bug No known bugs except for NYI items

9.58 layers/attention_layer.cpp File Reference

This is Attention Layer Class for Neural Network.

```
#include <attention_layer.h>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
```

Include dependency graph for attention_layer.cpp:



Namespaces

- nntrainer

Enumerations

- enum nntrainer::AttentionParams {
  query = 0, nntrainer::value = 1, **key** = 2, **weights**,
  query_fc_weight, **query_fc_bias**, **key_fc_weight**, **key_fc_bias**,
  value_fc_weight, **value_fc_bias**, **fc_weight**, **fc_bias**,
  projected_query, **projected_key**, **projected_value**, nntrainer::attention_weight,
  dropout_mask, **attention_output** }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.58.1 Detailed Description

This is Attention Layer Class for Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

1 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

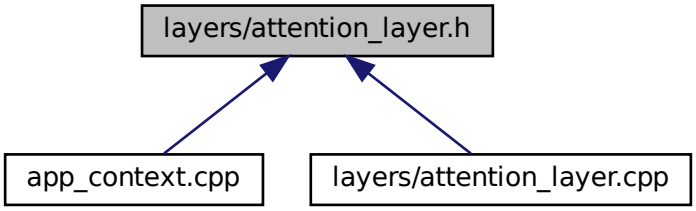
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.59 layers/attention_layer.h File Reference

This is Attention Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.59.1 Detailed Description

This is Attention Layer Class for Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

1 October 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

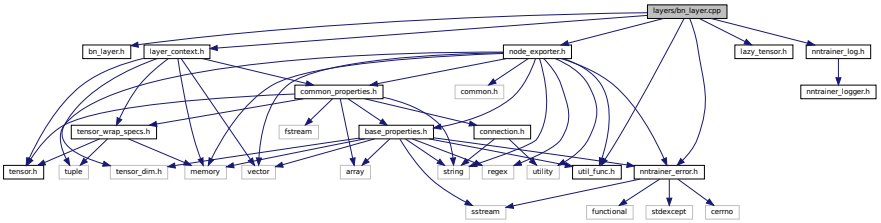
Bug No known bugs except for NYI items

9.60 layers/bn_layer.cpp File Reference

This is Batch Normalization Layer Class for Neural Network.

```
#include <bn_layer.h>
#include <layer_context.h>
#include <lazy_tensor.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for bn_layer.cpp:



Namespaces

- nntrainer

Enumerations

- enum **BNParams** {
  mu, **var**, **gamma**, **beta**,
  deviation, **invstd**, **cvar**, **t_reduced**,
  t_full }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.60.1 Detailed Description

This is Batch Normalization Layer Class for Neural Network.

Date

14 May 2020

See also

https://github.com/nnstreamer/nntrainer

Author

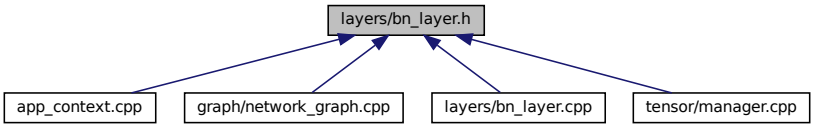
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.61 layers/bn_layer.h File Reference

This is Batch Normalization Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.61.1 Detailed Description

This is Batch Normalization Layer Class of Neural Network.

Date

14 May 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

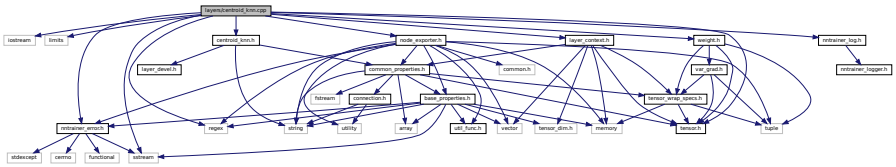
Bug No known bugs except for NYI items

9.62 layers/centroid_knn.cpp File Reference

This file contains the simple nearest neighbor layer.

```
#include <iostream>
#include <limits>
#include <regex>
#include <sstream>
#include <centroid_knn.h>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <tensor.h>
#include <weight.h>
```

Include dependency graph for centroid_knn.cpp:



Namespaces

- nntrainer

Enumerations

- enum **KNNParams** { **map**, **num_samples** }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.62.1 Detailed Description

This file contains the simple nearest neighbor layer.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 Jan 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

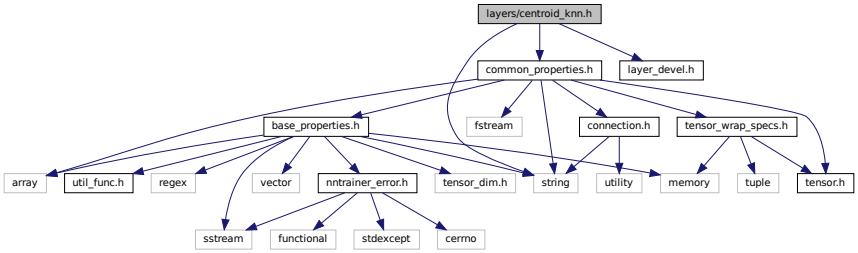
Bug No known bugs except for NYI items

This layer takes centroid and calculate l2 distance

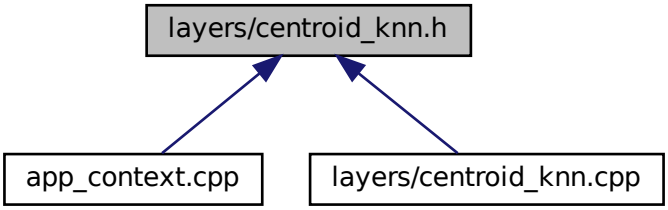
9.63 layers/centroid_knn.h File Reference

```
#include <string>
#include <common_properties.h>
#include <layer_devel.h>
```

Include dependency graph for centroid_knn.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::CentroidKNN

  Centroid KNN layer which takes centroid and do k-nearest neighbor classification.

Namespaces

- nntrainer

9.63.1 Detailed Description

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 Jan 2021

This file contains the simple nearest neighbor layer, this layer takes centroid and calculate l2 distance

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

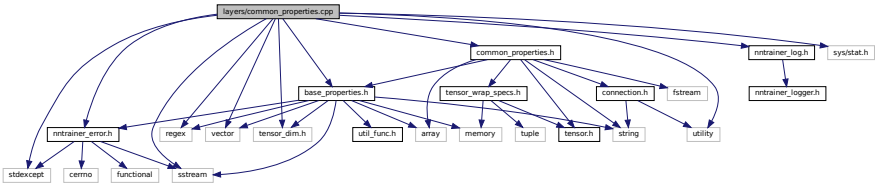
Bug No known bugs except for NYI items

9.64 layers/common_properties.cpp File Reference

This file contains implementation of common properties widely used across layers.

```
#include <base_properties.h>
#include <common_properties.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <stdexcept>
#include <tensor_dim.h>
#include <regex>
#include <sstream>
#include <sys/stat.h>
#include <utility>
#include <vector>
```

Include dependency graph for common_properties.cpp:



Classes

- class nntrainer::props::Padding_

  unsigned integer property, internally used to parse padding values

Namespaces

- nntrainer

9.64.1 Detailed Description

This file contains implementation of common properties widely used across layers.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

14 May 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

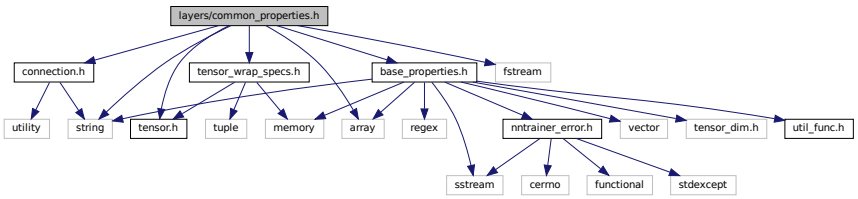
Bug No known bugs except for NYI items

9.65 layers/common_properties.h File Reference

This file contains list of common properties widely used across layers.

```
#include <array>
#include <fstream>
#include <string>
#include <base_properties.h>
#include <connection.h>
#include <tensor.h>
#include <tensor_wrap_specs.h>
```

Include dependency graph for common_properties.h:



This graph shows which files directly or indirectly include this file:

Classes

- class nntrainer::props::Name

  Name property, name is an identifier of an object.
- class nntrainer::props::Unit

  unit property, unit is used to measure how many weights are there
- class nntrainer::props::Trainable

  trainable property, use this to set and check how if certain layer is trainable
- class nntrainer::props::DisableBias

  DisableBias to disable the bias.
- class nntrainer::props::IntegrateBias

  Integrate bias_ih and bias_hh to bias_h to use only 1 bias (Used in rnn variant)
- class nntrainer::props::Normalization

  Normalization property, normalize the input to be in range [0, 1] if true.
- class nntrainer::props::Standardization

  Standardization property, standardization standardize the input to be mean 0 and std 1 if true.
- struct nntrainer::props::connection_prop_tag

  Connection prop tag type.
- class nntrainer::props::InputConnection

  InputSpec property, this defines connection specification of an input.
- class nntrainer::props::Epsilon

  Epsilon property, this is used to avoid divide by zero.
- class nntrainer::props::Momentum

  Momentum property, moving average in batch normalization layer.
- class nntrainer::props::SplitNumber

  split number property, split number indicates how many numbers of outs are generated by spliting the input dimension
- class nntrainer::props::Axis

  Axis property, idx in the dimension.
- class nntrainer::props::SplitDimension

  SplitDimension property, dimension along which to split the input.
- class nntrainer::props::ConcatDimension

  ConcatDimension property, dimension along which to concat the input.
- class nntrainer::props::ReduceDimension

  ReduceDimension property, dimension along which to reduce the input.
- class nntrainer::props::FilterSize

  FilterSize property, filter size is used to measure how many filters are there.
- class nntrainer::props::KernelSize

  KernelSize property, kernel size is used to measure the filter size.
- class nntrainer::props::PoolSize

  PoolSize property, pool size is used to measure the pooling size.
- class nntrainer::props::Stride

  Stride property, stride is used to measure how much it will be slide the filter.
- class nntrainer::props::Dilation

  Dilation property, dilation indicates how many space will be inserted between kernel element.
- class nntrainer::props::Padding2D

  Padding2D property, this is used to calculate padding2D.
- class nntrainer::props::Padding1D

  Padding1D property, this is used to calculate padding2D.
- class nntrainer::props::InDim

  InDim property, in dim is the size of vocabulary in the text data.
- class nntrainer::props::OutDim

OutDim property, out dim is the size of the vector space in which words will be embedded.

- class nntrainer::props::ZeroIdxMask

  Zero idx mask property for embedding where the value of embedding will be zero.
- class nntrainer::props::DropOutRate

  DropOutRate property, this defines drop out specification of layer.
- class nntrainer::props::RandomTranslate

  TranslationFactor property, this defines how far the image is translated.
- class nntrainer::props::FilePath

  Props containing file path value.
- class nntrainer::props::DirPath

  Props containing directory path value.
- class nntrainer::props::ReturnSequences

  return sequence property, used to check whether return only the last output. Return last output if true.
- class nntrainer::props::Bidirectional

  bidirectional property, used to make bidirectional layers
- class nntrainer::props::AsSequence

  Identifiers to locate a connection which should be returned as whole used in recurrent realizer.
- class nntrainer::props::InputIsSequence

  *Identifiers to locate an **input** connection which should be sequenced for the connection.*
- class nntrainer::props::ResetAfter

  ResetAfter property, apply reset gate after matrix multiplication if this property is true. Apply before the multiplication if false. Used in gru, grucell.
- class nntrainer::props::NumClass

  Number of class.
- class nntrainer::props::BasicRegularizerConstant

  BasicRegularizerConstant property, this defines how much regularize the weight.
- class nntrainer::props::WeightRegularizerConstant

  WeightRegularizerConstant property, this defines how much regularize the weight.
- class nntrainer::props::WeightDecay

  WeightDecay property, this defines how much to decay the weight.
- class nntrainer::props::BiasDecay

  BiasDecay property, this defines how much regularize the weight.
- class nntrainer::props::OutputLayer

  Output Layer name property which saves a single connection (practically, std::vector<InputLayers> is used)
- class nntrainer::props::LabelLayer

  label Layer name property which saves a single connection (practically, std::vector<LabelLayers> is used)
- struct nntrainer::props::ActivationTypeInfo

  Enumeration of activation function type.
- class nntrainer::props::Activation

  Activation Enumeration Information.
- class nntrainer::props::HiddenStateActivation

  HiddenStateActivation Enumeration Information.
- class nntrainer::props::RecurrentActivation

  RecurrentActivation Enumeration Information.
- struct nntrainer::props::InitializerInfo

  Enumeration of tensor initialization type.
- class nntrainer::props::WeightInitializer

  WeightInitializer Initialization Enumeration Information.
- class nntrainer::props::BiasInitializer

  BiasInitializer Initialization Enumeration Information.
- class nntrainer::props::BNPARAMS_MU_INIT

BNPARAMS_MU_INIT *Initialization Enumeration Information.*

- class nntrainer::props::BNPARAMS_VAR_INIT

  BNPARAMS_VAR_INIT *Initialization Enumeration Information.*
- class nntrainer::props::BNPARAMS_GAMMA_INIT

  BNPARAMS_GAMMA_INIT *Initialization Enumeration Information.*
- class nntrainer::props::BNPARAMS_BETA_INIT

  BNPARAMS_BETA_INIT *Initialization Enumeration Information.*
- struct nntrainer::props::RegularizerInfo

  Enumeration of tensor regularization type.
- class nntrainer::props::BasicRegularizer

  BasicRegularizer *Regularization Enumeration Information.*
- class nntrainer::props::WeightRegularizer

  WeightRegularizer *Regularization Enumeration Information.*
- struct nntrainer::props::PoolingTypeInfo

  Enumeration of pooling type.
- class nntrainer::props::PoolingType

  Pooling Type Enumeration Information.
- struct nntrainer::props::FlipDirectionInfo

  Enumeration of flip direction.
- class nntrainer::props::FlipDirection

  FlipDirection *Enumeration Information.*
- class nntrainer::props::Timestep

  timestep property, timestep is used to identify for which timestep should the lstm/gru/rnn layer do the operation for
- class nntrainer::props::MaxTimestep

  maximum timestep property, timestep is used to identify for the maximum time unroll possible for lstm/gru/rnn layer
- class nntrainer::props::GenericShape

  generic shape property which saves a single tensor shape (practically, std::array<GenericShape> is used)
- class nntrainer::props::TargetShape

  target shape property which saves a single tensor shape (practically, std::array<TargetShape> is used)
- class nntrainer::props::MoL_K

  K property, K is the size of the three projections in MoL attention.
- class nntrainer::props::NumHeads

  NumHeads *property, NumHeads is number of head in multi head attention.*
- class nntrainer::props::ProjectedKeyDim

  ProjectedKeyDim *property, projected key dim per head in multi head attention.*
- class nntrainer::props::ProjectedValueDim

  ProjectedValueDim *property, projected value dim per head in multi head attention.*
- class nntrainer::props::OutputShape

  OutputShape *property, output shape of multi head attention.*
- struct nntrainer::props::ReturnAttentionWeightInfo

  Enumeration of return attention weight.
- class nntrainer::props::ReturnAttentionWeight

  ReturnAttentionWeight, return attention weight.
- class nntrainer::props::AverageAttentionWeight

  AverageAttentionWeight, average attention weight.
- class nntrainer::props::ClipGradByGlobalNorm

  properties for getting the clipping value to clip the gradient by norm
- class nntrainer::props::LearningRate

  Learning Rate props.
- class nntrainer::props::Iteration

  Iteration *props.*

- class nntrainer::props::DecayRate

  Decay rate property.
- class nntrainer::props::DecaySteps

  decay steps property
- class nntrainer::props::PropsUserData

  User data props.

Namespaces

- nntrainer

Enumerations

- enum nntrainer::ActivationType { nntrainer::ActivationType::ACT_TANH, nntrainer::ActivationType::ACT_SIGMOID, nntrainer::ActivationType::ACT_RELU, nntrainer::ActivationType::ACT_SWISH, nntrainer::ActivationType::ACT_SOFTMAX, nntrainer::ActivationType::ACT_LEAKY_RELU, nntrainer::ActivationType::ACT_NO nntrainer::ActivationType::ACT_UNKNOWN }

  Enumeration of activation function type.

9.65.1 Detailed Description

This file contains list of common properties widely used across layers.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

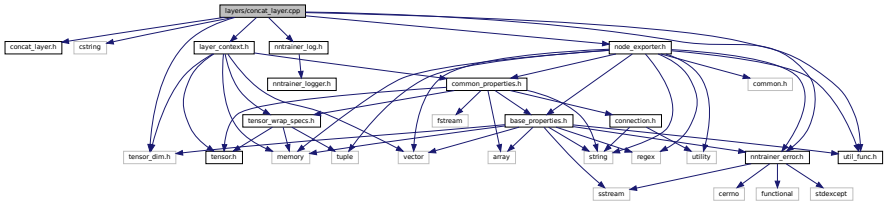
Bug No known bugs except for NYI items

9.66 layers/concat_layer.cpp File Reference

This is Concat Layer Class for Neural Network.

```
#include <concat_layer.h>
#include <cstring>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <tensor_dim.h>
#include <util_func.h>
```

Include dependency graph for concat_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.66.1 Detailed Description

This is Concat Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

27 Oct 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

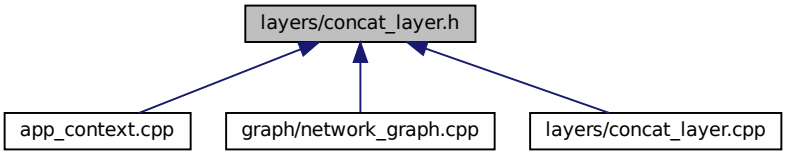
Bug No known bugs except for NYI items

Todo merge concat and split layer to a common implementation

9.67 layers/concat_layer.h File Reference

This is Concat Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.67.1 Detailed Description

This is Concat Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

27 Oct 2020

See also

https://github.com/nnstreamer/nntrainer

Author

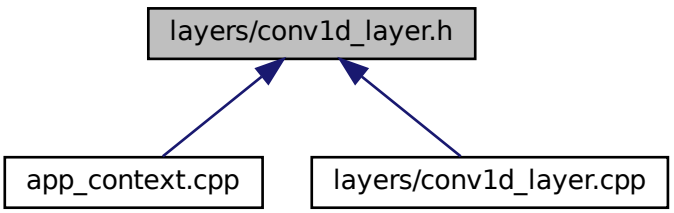
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.68 layers/conv1d_layer.h File Reference

This is Convolution 1D Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.68.1 Detailed Description

This is Convolution 1D Layer Class for Neural Network.

Copyright (C) 2021 Jijoong Moon jijoong.moon@samsung.com

Date

13 Oct 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2021 Jijoong Moon jijoong.moon@samsung.com

Date

13 Oct 2021

See also

https://github.com/nnstreamer/nntrainer

Author

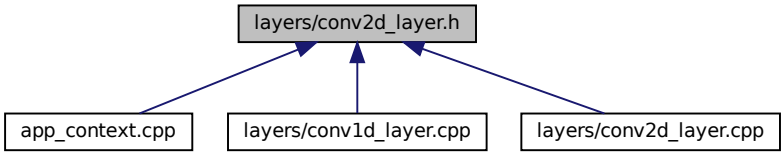
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.69 layers/conv2d_layer.h File Reference

This is Convolution Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.69.1 Detailed Description

This is Convolution Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

02 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com
Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

01 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

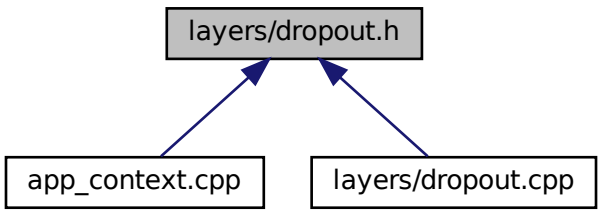
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.70 layers/dropout.h File Reference

This is DropOut Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.70.1 Detailed Description

This is DropOut Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

05 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

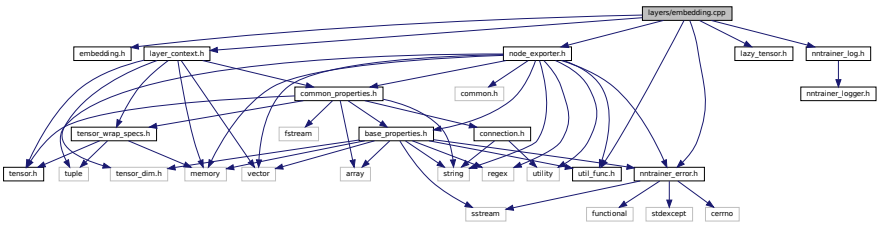
Bug No known bugs except for NYI items

9.71 layers/embedding.cpp File Reference

This is Embedding Layer Class of Neural Network.

```
#include <embedding.h>
#include <layer_context.h>
#include <lazy_tensor.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for embedding.cpp:



Namespaces

- nntrainer

Enumerations

- enum **EmbeddingParams** { **weight** }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.71.1 Detailed Description

This is Embedding Layer Class of Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

04 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

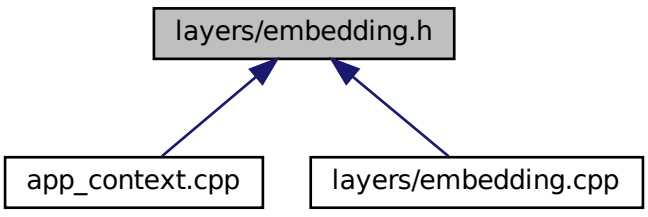
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.72 layers/embedding.h File Reference

This is Embedding Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.72.1 Detailed Description

This is Embedding Layer Class of Neural Network.

Copyright (C) 2021 Jijoong Moon jijoong.moon@samsung.com

Date

04 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

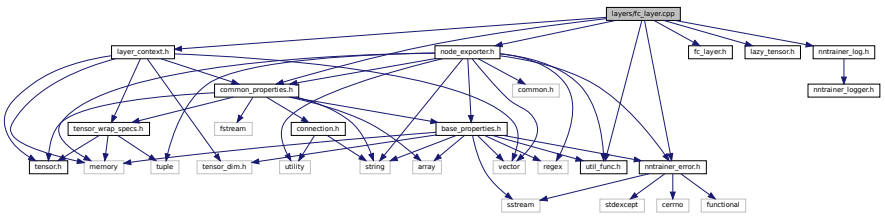
Bug No known bugs except for NYI items

9.73 layers/fc_layer.cpp File Reference

This is Fully Connected Layer Class for Neural Network.

```
#include <common_properties.h>
#include <fc_layer.h>
#include <layer_context.h>
#include <lazy_tensor.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for fc_layer.cpp:



Namespaces

- nntrainer

Enumerations

- enum **FCParams** { **weight**, **bias** }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.73.1 Detailed Description

This is Fully Connected Layer Class for Neural Network.

Date

14 May 2020

See also

https://github.com/nnstreamer/nntrainer

Author

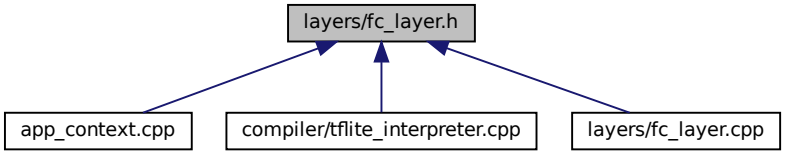
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.74 layers/fc_layer.h File Reference

This is Fully Connected Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.74.1 Detailed Description

This is Fully Connected Layer Class of Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

14 May 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

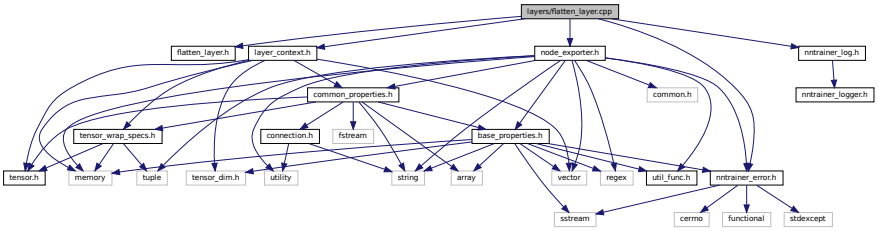
Bug No known bugs except for NYI items

9.75 layers/flatten_layer.cpp File Reference

This is Flatten Layer Class for Neural Network.

```
#include <flatten_layer.h>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
```

Include dependency graph for flatten_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.75.1 Detailed Description

This is Flatten Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

16 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

Todo Update flatten to work in-place properly.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

16 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

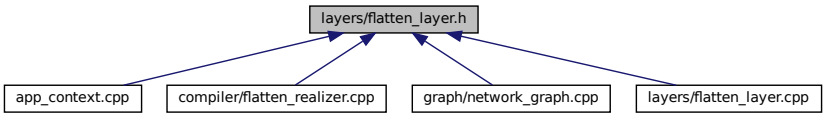
Bug No known bugs except for NYI items

Todo Update flatten to work in-place properly.

9.76 layers/flatten_layer.h File Reference

This is Flatten Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.76.1 Detailed Description

This is Flatten Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

16 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

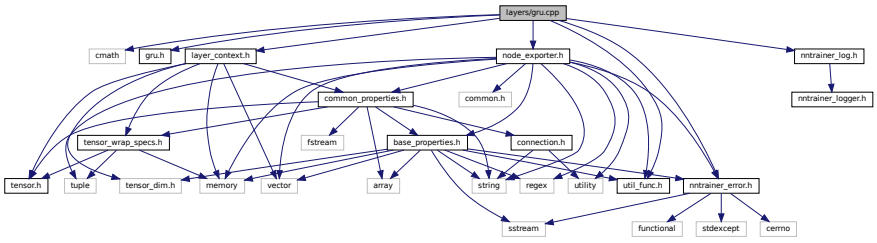
Bug No known bugs except for NYI items

9.77 layers/gru.cpp File Reference

This is Gated Recurrent Unit Layer Class of Neural Network.

```
#include <cmath>
#include <gru.h>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for gru.cpp:



Namespaces

- nntrainer

Enumerations

- enum **GRUParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **hidden_state**, **zrg**, **h_prev**,
  dropout_mask }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.77.1 Detailed Description

This is Gated Recurrent Unit Layer Class of Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

17 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

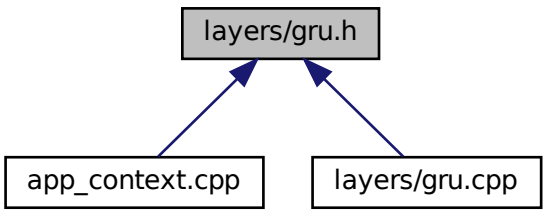
Bug No known bugs except for NYI items

h_prev -------—d1------—>[∗]------—d0-—>[+]—d0--> h dh_nx | | | | d0 dh | d14 | d2 d3 | | | +-—[1-]-—>[∗] | [∗]<—+ d15 |d5 | d6 | | |rt | zt |gt | | [sig] [sig] [tanh] | | |d16 | d7 |d8 | | [+] [+] [+] | | / \d16 | \ d7 / \ d8 | | Whhr Wxhr Whhz Wxhz Whhg Wxhg | | |d17 |d13 |d12 |d11 |d10 | d9 +- |–+-—|—+ | | | +-------—|-------|--------—+ | xs----------—+------—+----------—+

9.78 layers/gru.h File Reference

This is Gated Recurrent Unit Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.78.1 Detailed Description

This is Gated Recurrent Unit Layer Class of Neural Network.

Copyright (C) 2021 Jijoong Moon jijoong.moon@samsung.com

Date

31 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

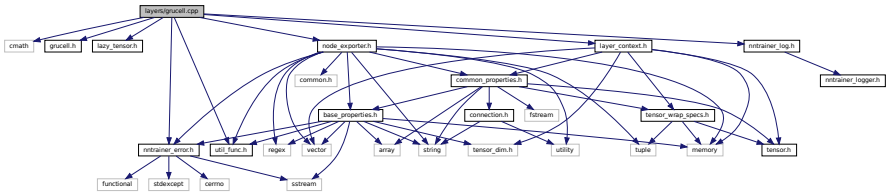
Bug No known bugs except for NYI items

9.79 layers/grucell.cpp File Reference

This is Gated Recurrent Unit Cell Layer Class of Neural Network.

```
#include <cmath>
#include <grucell.h>
#include <lazy_tensor.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <util_func.h>
#include <layer_context.h>
```

Include dependency graph for grucell.cpp:



Namespaces

- nntrainer

Enumerations

- enum **GRUCellParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **zrg**, **dropout_mask** }

Functions

- static void nntrainer::grucell_forwarding (const unsigned int unit, const unsigned int batch_size, const bool disable_bias, const bool integrate_bias, const bool reset_after, ActiFunc &acti_func, ActiFunc &recurrent↩ _acti_func, const Tensor &input, const Tensor &prev_hidden_state, Tensor &hidden_state, const Tensor &weight_ih, const Tensor &weight_hh, const Tensor &bias_h, const Tensor &bias_ih, const Tensor &bias_hh, Tensor &zrg)

  gru forwarding
- static void nntrainer::grucell_calcGradient (const unsigned int unit, const unsigned int batch_size, const bool disable_bias, const bool integrate_bias, const bool reset_after, ActiFunc &acti_func, ActiFunc &recurrent_↩ acti_func, const Tensor &input, const Tensor &prev_hidden_state, Tensor &d_prev_hidden_state, const Tensor &d_hidden_state, Tensor &d_weight_ih, const Tensor &weight_hh, Tensor &d_weight_hh, Tensor &d_↩ bias_h, Tensor &d_bias_ih, const Tensor &bias_hh, Tensor &d_bias_hh, const Tensor &zrg, Tensor &d_zrg)

  gru calcGradient

9.79.1 Detailed Description

This is Gated Recurrent Unit Cell Layer Class of Neural Network.

Copyright (C) 2021 hyeonseok lee hs89.lee@samsung.com

Date

28 Oct 2021

See also

https://github.com/nnstreamer/nntrainer

Author

hyeonseok lee hs89.lee@samsung.com

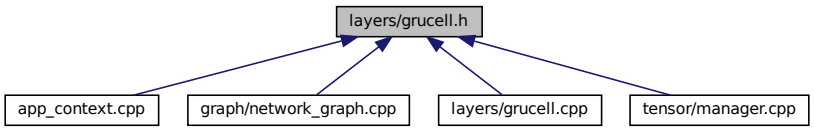
Bug No known bugs except for NYI items

h_prev -------d1------>[∗]------d0---->[+]—d0--> h d_h_prev | | | | d0 dh | d14 | d2 d3 | | | +----[1-]----->[∗] | [∗]<—+ d15 |d5 | d6 | | |reset_g| update_gate | memory_cell | | [sig] [sig] [tanh] | | |d16 | d7 |d8 | | [+] [+] [+] | | / \d16 | \ d7 / \ d8 | | Whhr Wxhr Whhz Wxhz Whhg Wxhg | | |d17 |d13 |d12 |d11 |d10 | d9 +- |–+-----|—+ | | | +--------|-------|--------+ | xs----------------+-------+----------+

9.80 layers/grucell.h File Reference

This is Gated Recurrent Unit Cell Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.80.1 Detailed Description

This is Gated Recurrent Unit Cell Layer Class of Neural Network.

Copyright (C) 2021 hyeonseok lee hs89.lee@samsung.com

Date

28 Oct 2021

See also

https://github.com/nnstreamer/nntrainer

Author

hyeonseok lee hs89.lee@samsung.com

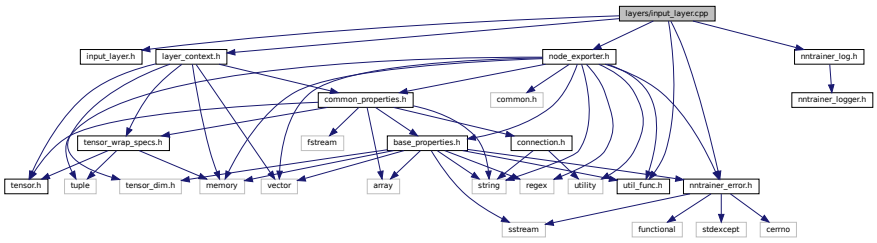
Bug No known bugs except for NYI items

9.81 layers/input_layer.cpp File Reference

This is Input Layer Class for Neural Network.

```
#include <input_layer.h>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for input_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.81.1 Detailed Description

This is Input Layer Class for Neural Network.

Date

14 May 2020

See also

https://github.com/nnstreamer/nntrainer

Author

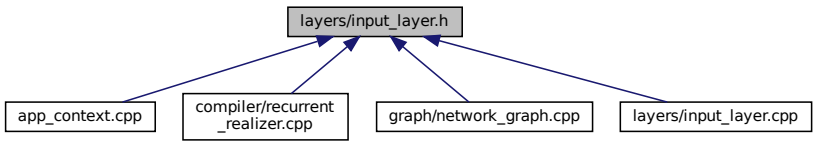
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.82 layers/input_layer.h File Reference

This is Input Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.82.1 Detailed Description

This is Input Layer Class of Neural Network.

Date

14 May 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

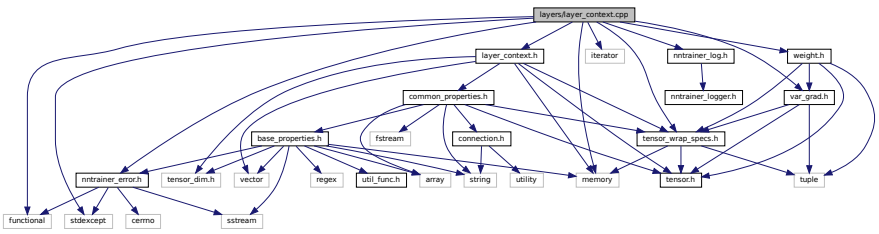
Bug No known bugs except for NYI items

9.83 layers/layer_context.cpp File Reference

This is the layer context for each layer.

```
#include "nntrainer_error.h"
#include <functional>
#include <memory>
#include <tensor_wrap_specs.h>
#include <iterator>
#include <layer_context.h>
#include <nntrainer_log.h>
#include <stdexcept>
#include <var_grad.h>
#include <weight.h>
```

Include dependency graph for layer_context.cpp:



Namespaces

- nntrainer

Functions

- static void nntrainer::suffixSpec (VarGradSpecV2 &spec, unsigned int idx)
  rename specification

9.83.1 Detailed Description

This is the layer context for each layer.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

26 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

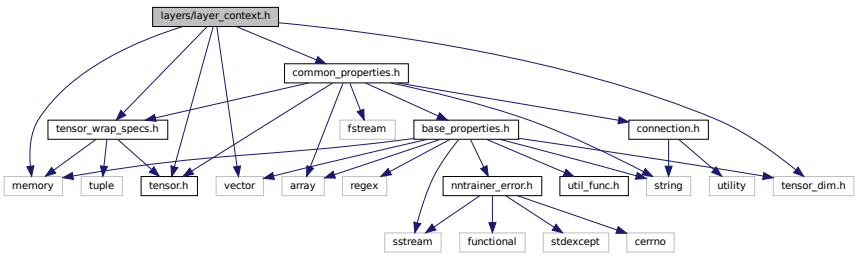
Bug No known bugs except for NYI items

9.84 layers/layer_context.h File Reference

This is the layer context for each layer.

```
#include <memory>
#include <vector>
#include <common_properties.h>
#include <tensor.h>
#include <tensor_dim.h>
#include <tensor_wrap_specs.h>
```

Include dependency graph for layer_context.h:



This graph shows which files directly or indirectly include this file:

Classes

- class nntrainer::InitLayerContext
- class nntrainer::RunLayerContext

Namespaces

- nntrainer

9.84.1 Detailed Description

This is the layer context for each layer.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

10 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

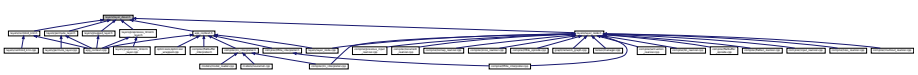
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.85 layers/layer_devel.h File Reference

This is Layer classes of Neural Network.

This graph shows which files directly or indirectly include this file:



9.85.1 Detailed Description

This is Layer classes of Neural Network.

Date

10 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

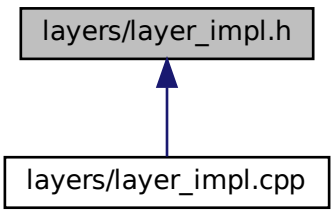
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.86 layers/layer_impl.h File Reference

This is base layer implementation class.

This graph shows which files directly or indirectly include this file:



9.86.1 Detailed Description

This is base layer implementation class.

This is base Optimizer implementation class.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

21 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

21 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

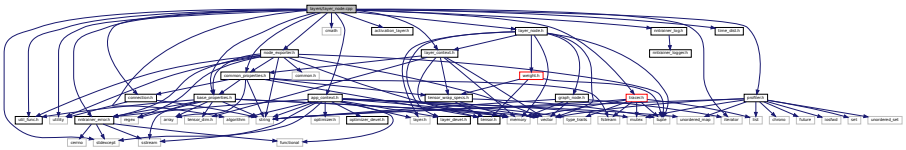
LayerImpl forms the base class for all the layer with weights and bias parameters. LayerImpl provides parsing of properties like Weight/bias initializer and regularizers. LayerImpl also provides checks for double calls to finalize function.

9.87 layers/layer_node.cpp File Reference

This is the layer node for network graph.

```
#include "layer_context.h"
#include <algorithm>
#include <cmath>
#include <iterator>
#include <stdexcept>
#include <utility>
#include <activation_layer.h>
#include <app_context.h>
#include <base_properties.h>
#include <common_properties.h>
#include <connection.h>
#include <layer_node.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <profiler.h>
#include <time_dist.h>
#include <tracer.h>
#include <util_func.h>
```

Include dependency graph for layer_node.cpp:



Classes

- class nntrainer::props::Flatten

  Flatten property, true if needs flatten layer afterwards.
- class nntrainer::props::Distribute

  Distribute property, true if it distribute across layer.
- class nntrainer::props::Loss

  Loss property, this defines loss specification of layer.
- class nntrainer::props::InputShape

  Input shape property which saves a single tensor shape (practically, std::array<InputShape> is used)
- class nntrainer::props::SharedFrom

  properties for shared from

Namespaces

- nntrainer

Enumerations

- enum nntrainer::PrintOption {
  nntrainer::PRINT_INST_INFO = (1 << 0), nntrainer::PRINT_SHAPE_INFO = (1 << 1), nntrainer::PRINT_PROP = (1 << 2), nntrainer::PRINT_PROP_META = (1 << 3),
  nntrainer::PRINT_WEIGHTS = (1 << 4), nntrainer::PRINT_METRIC = (1 << 5) }

  Print Options when printing layer info.

Functions

- std::unique_ptr< LayerNode > nntrainer::createLayerNode (const ml::train::LayerType &type, const std::vector< std::string > &properties)

  Layer factory creator with constructor.
- std::unique_ptr< LayerNode > nntrainer::createLayerNode (const std::string &type, const std::vector< std::string > &properties)

  Layer factory creator with constructor.
- std::unique_ptr< LayerNode > nntrainer::createLayerNode (std::unique_ptr< nntrainer::Layer > &&layer, const std::vector< std::string > &properties)

  Layer factory creator with constructor.
- std::ostream & **nntrainer::operator**<< (std::ostream &out, const LayerNode &l)

9.87.1 Detailed Description

This is the layer node for network graph.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

1 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

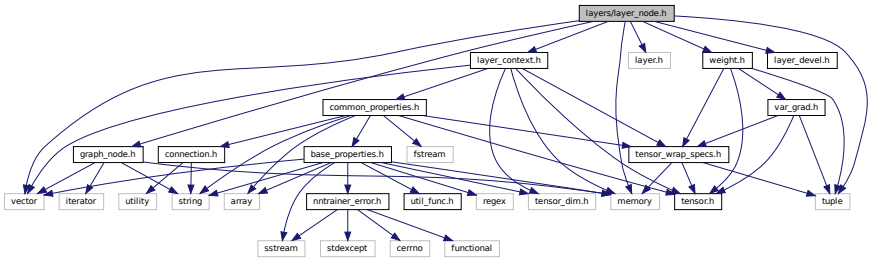
Bug No known bugs except for NYI items

9.88 layers/layer_node.h File Reference

This is the layer node for network graph.

```
#include <memory>
#include <tuple>
#include <vector>
#include <graph_node.h>
#include <layer.h>
#include <layer_context.h>
#include <layer_devel.h>
#include <weight.h>
```

Include dependency graph for layer_node.h:



This graph shows which files directly or indirectly include this file:

Classes

- class nntrainer::LayerNode

  layer node class for the graph

Namespaces

- nntrainer

Enumerations

- enum nntrainer::InPlace { nntrainer::InPlace::NONE, nntrainer::InPlace::RESTRICTING, nntrainer::InPlace::NON_RESTRICTIN
  }

  Enum class for the various types of inplace modes supported by layer.

Functions

- std::unique_ptr< LayerNode > nntrainer::createLayerNode (const ml::train::LayerType &type, const std::vector< std::string > &properties)

  Layer factory creator with constructor.

- std::unique_ptr< LayerNode > nntrainer::createLayerNode (const std::string &type, const std::vector< std::string > &properties)

  Layer factory creator with constructor.

- std::unique_ptr< LayerNode > nntrainer::createLayerNode (std::unique_ptr< nntrainer::Layer > &&layer, const std::vector< std::string > &properties)

  Layer factory creator with constructor.

9.88.1 Detailed Description

This is the layer node for network graph.

Date

1 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Todo Add printPreset support

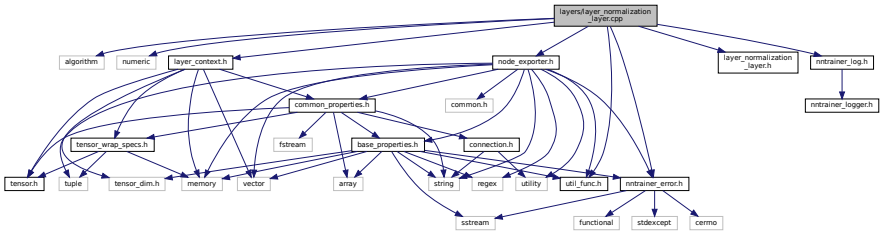
LayerNode provides a node wrapper around the Layer class to form a GraphNode. Each layer is wrapped with LayerNode in order to add it to a graph. Each LayerNode contains only 1 layer inside. LayerNode also intercepts certain properties of the layer which are either related to graph related connections (input_connections, output_connections, activation, flatten, distribute, name) or essential for the description of the layer (trainable, input_dims) itself. These properties, if needed by the layer object, are provided access to via LayerContext.

9.89 layers/layer_normalization_layer.cpp File Reference

This is Layer Normalization Layer Class for Neural Network.

```
#include <algorithm>
#include <numeric>
#include <layer_context.h>
#include <layer_normalization_layer.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for layer_normalization_layer.cpp:



Namespaces

- nntrainer

Enumerations

- enum **LNParams** {
  gamma, **beta**, **deviation**, **variance**,
  inv_std_dev, **temp_origin_size**, **temp_normalized_size** }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.89.1 Detailed Description

This is Layer Normalization Layer Class for Neural Network.

Copyright (C) 2022 hyeonseok Lee hs89.lee@samsung.com

Date

25 July 2022

See also

https://github.com/nnstreamer/nntrainer https://arxiv.org/abs/1607.↩
06450

Author

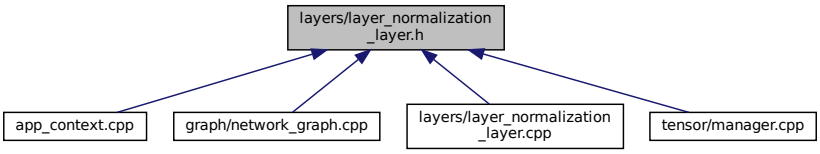
hyeonseok Lee hs89.lee@samsung.com

Bug No known bugs except for NYI items

9.90 layers/layer_normalization_layer.h File Reference

This is Layer Normalization Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.90.1 Detailed Description

This is Layer Normalization Layer Class for Neural Network.

Copyright (C) 2022 hyeonseok Lee hs89.lee@samsung.com

Date

25 July 2022

See also

https://github.com/nnstreamer/nntrainer https://arxiv.org/abs/1607.↩
06450

Author

hyeonseok Lee hs89.lee@samsung.com

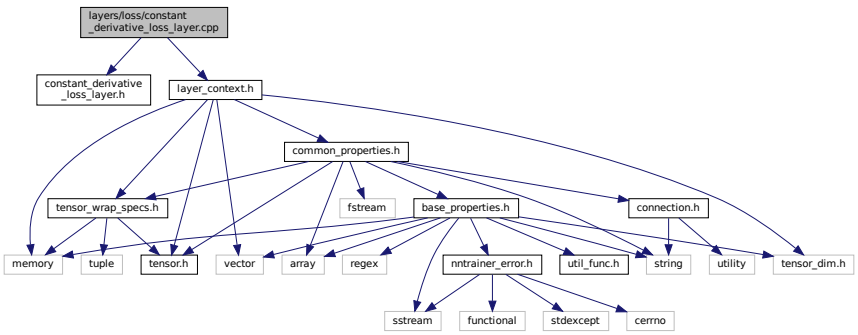
Bug No known bugs except for NYI items

9.91 layers/loss/constant_derivative_loss_layer.cpp File Reference

This patch contains constant derivative loss implementation.

```
#include <constant_derivative_loss_layer.h>
#include <layer_context.h>
```

Include dependency graph for constant_derivative_loss_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr int **nntrainer::SINGLE_INOUT_IDX** = 0

9.91.1 Detailed Description

This patch contains constant derivative loss implementation.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

05 Oct 2021

Note

This is special type of loss to feed an arbitrary derivative value to the last layer.

See also

https://github.com/nnstreamer/nntrainer

Author

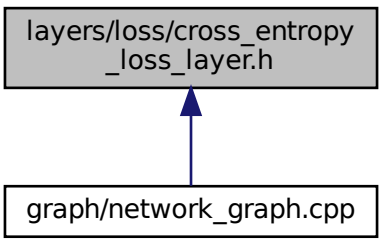
Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

9.92 layers/loss/cross_entropy_loss_layer.h File Reference

This is Cross Entropy Loss Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.92.1 Detailed Description

This is Cross Entropy Loss Layer Class of Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

24 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

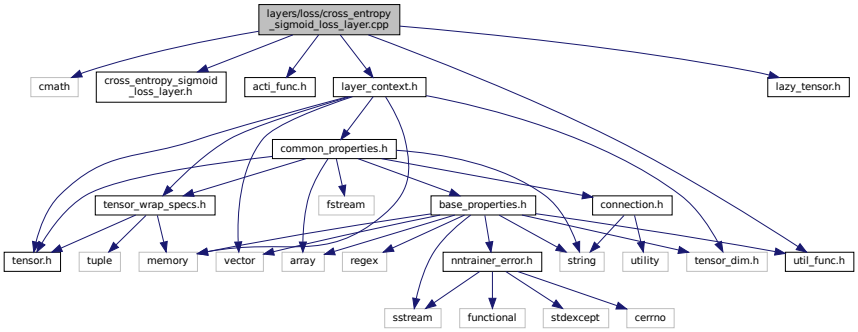
Bug No known bugs except for NYI items

9.93 layers/loss/cross_entropy_sigmoid_loss_layer.cpp File Reference

This is MSE Loss Layer Class of Neural Network.

```
#include <cmath>
#include <cross_entropy_sigmoid_loss_layer.h>
#include <acti_func.h>
#include <layer_context.h>
#include <lazy_tensor.h>
#include <util_func.h>
```

Include dependency graph for cross_entropy_sigmoid_loss_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.93.1 Detailed Description

This is MSE Loss Layer Class of Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

24 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

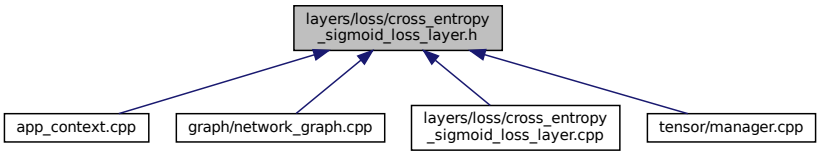
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.94 layers/loss/cross_entropy_sigmoid_loss_layer.h File Reference

This is Cross Entropy Sigmoid with Sigmoid Loss Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.94.1 Detailed Description

This is Cross Entropy Sigmoid with Sigmoid Loss Layer Class of Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

24 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

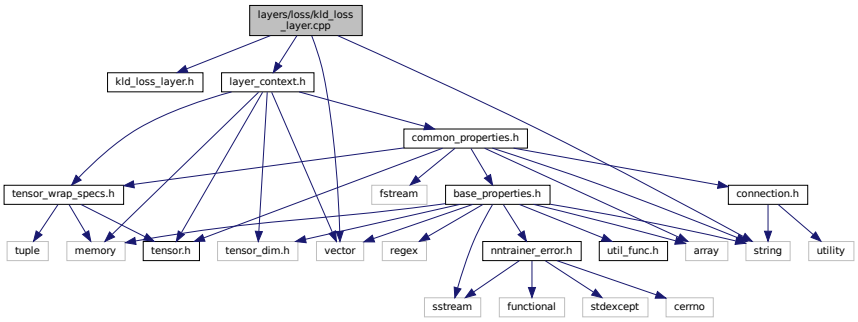
Bug No known bugs except for NYI items

9.95 layers/loss/kld_loss_layer.cpp File Reference

KLD (Kullback-Leibler Divergence) loss implementation.

```
#include <kld_loss_layer.h>
#include <layer_context.h>
#include <string>
#include <vector>
```

Include dependency graph for kld_loss_layer.cpp:



Namespaces

- nntrainer

9.95.1 Detailed Description

KLD (Kullback-Leibler Divergence) loss implementation.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

25 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

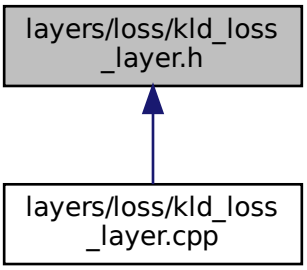
Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

9.96 layers/loss/kld_loss_layer.h File Reference

KLD (Kullback-Leibler Divergence) loss implementation.

This graph shows which files directly or indirectly include this file:



9.96.1 Detailed Description

KLD (Kullback-Leibler Divergence) loss implementation.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

25 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

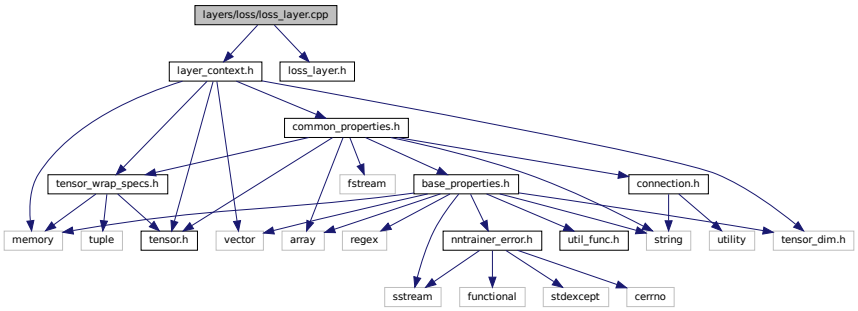
Bug No known bugs except for NYI items

9.97 layers/loss/loss_layer.cpp File Reference

This is Loss Layer Class for Neural Network.

```
#include <layer_context.h>
#include <loss_layer.h>
```

Include dependency graph for loss_layer.cpp:



Namespaces

- nntrainer

9.97.1 Detailed Description

This is Loss Layer Class for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

12 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

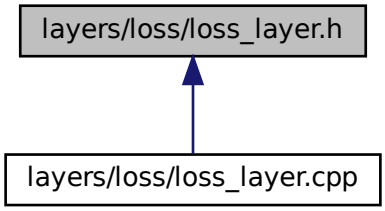
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.98 layers/loss/loss_layer.h File Reference

This is Loss Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.98.1 Detailed Description

This is Loss Layer Class of Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

12 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

12 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

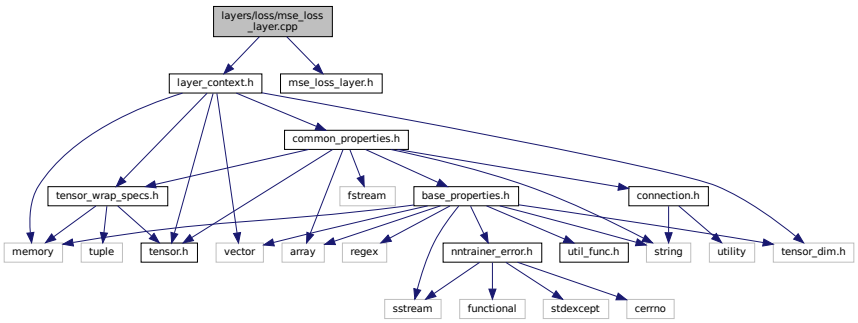
Bug No known bugs except for NYI items

9.99 layers/loss/mse_loss_layer.cpp File Reference

This is MSE Loss Layer Class of Neural Network.

```
#include <layer_context.h>
#include <mse_loss_layer.h>
```

Include dependency graph for mse_loss_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.99.1 Detailed Description

This is MSE Loss Layer Class of Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

24 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

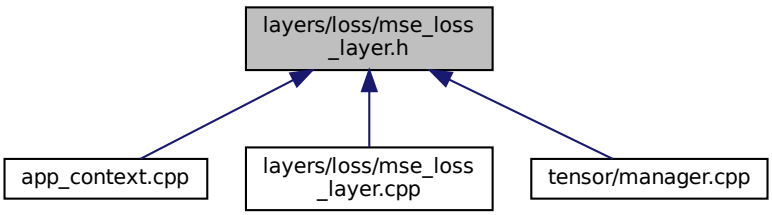
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.100 layers/loss/mse_loss_layer.h File Reference

This is MSE Loss Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.100.1 Detailed Description

This is MSE Loss Layer Class of Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

24 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

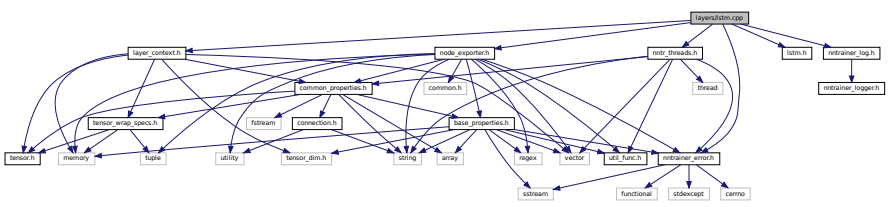
Bug No known bugs except for NYI items

9.101 layers/lstm.cpp File Reference

This is Long Short-Term Memory Layer Class of Neural Network.

```
#include <layer_context.h>
#include <lstm.h>
#include <nntr_threads.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
```

Include dependency graph for lstm.cpp:



Namespaces

- nntrainer

Enumerations

- enum **LSTMParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **hidden_state**, **cell_state**, **ifgo**,
  reverse_weight_ih, **reverse_weight_hh**, **reverse_bias_h**, **reverse_bias_ih**,
  reverse_bias_hh, **reverse_hidden_state**, **reverse_cell_state**, **reverse_ifgo**,
  dropout_mask }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.101.1 Detailed Description

This is Long Short-Term Memory Layer Class of Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

17 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

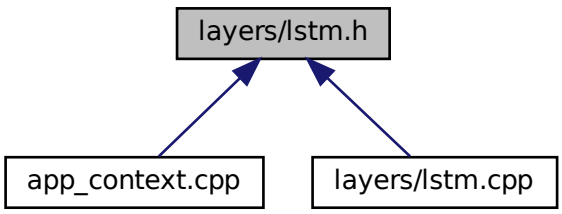
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.102 layers/lstm.h File Reference

This is Long Short-Term Memory Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.102.1 Detailed Description

This is Long Short-Term Memory Layer Class of Neural Network.

Copyright (C) 2021 Jijoong Moon jijoong.moon@samsung.com

Date

31 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

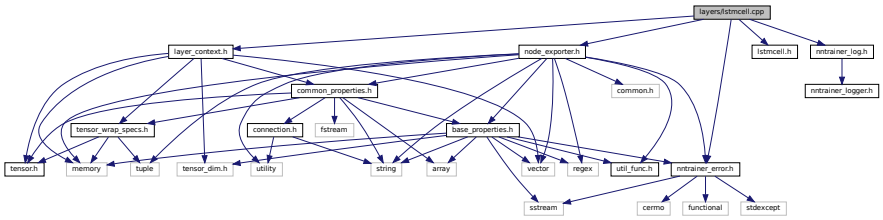
Bug No known bugs except for NYI items

9.103 layers/lstmcell.cpp File Reference

This is LSTMCell Layer Class of Neural Network.

```
#include <layer_context.h>
#include <lstmcell.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
```

Include dependency graph for lstmcell.cpp:



Namespaces

- nntrainer

Enumerations

- enum **LSTMCellParams** { **weight_ih**, **weight_hh**, **bias_h**, **bias_ih**, **bias_hh**, **ifgo**, **dropout_mask** }

9.103.1 Detailed Description

This is LSTMCell Layer Class of Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

17 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

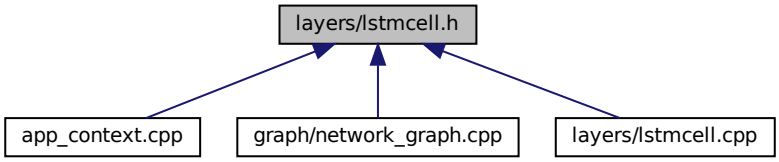
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.104 layers/lstmcell.h File Reference

This is LSTMCell Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.104.1 Detailed Description

This is LSTMCell Layer Class of Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

31 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

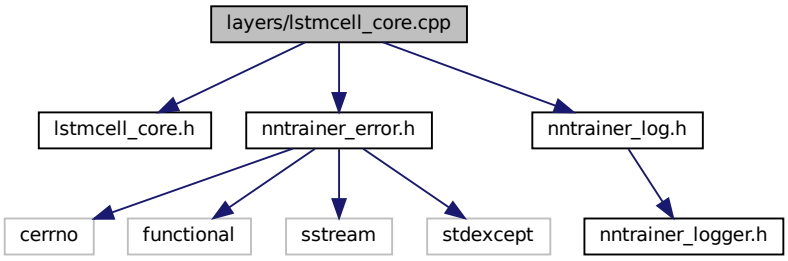
Bug No known bugs except for NYI items

9.105 layers/lstmcell_core.cpp File Reference

This is lstm core class.

```
#include <lstmcell_core.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
```

Include dependency graph for lstmcell_core.cpp:



Namespaces

- nntrainer

9.105.1 Detailed Description

This is lstm core class.

Copyright (C) 2021 hyeonseok lee hs89.lee@samsung.com

Date

25 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

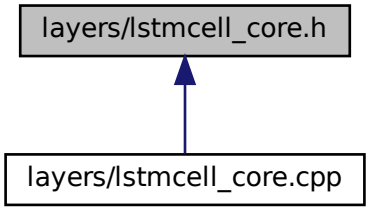
hyeonseok lee hs89.lee@samsung.com

Bug No known bugs except for NYI items

9.106 layers/lstmcell_core.h File Reference

This is lstm core class.

This graph shows which files directly or indirectly include this file:



9.106.1 Detailed Description

This is lstm core class.

Copyright (C) 2021 hyeonseok lee hs89.lee@samsung.com

Date

25 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

hyeonseok lee hs89.lee@samsung.com

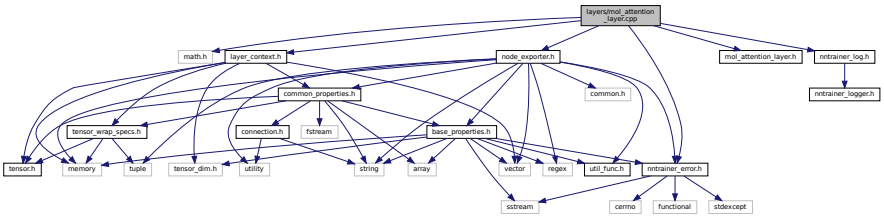
Bug No known bugs except for NYI items

9.107 layers/mol_attention_layer.cpp File Reference

This is MoL Attention Layer Class for Neural Network.

```
#include <math.h>
#include <layer_context.h>
#include <mol_attention_layer.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
```

Include dependency graph for mol_attention_layer.cpp:



Namespaces

- nntrainer

Enumerations

- enum nntrainer::MoLAttentionParams {
  query = 0, nntrainer::value = 1, **state** = 2, **mask_len** = 3,
  fc_w, **fc_bias**, **fc_proj_w**, **fc_out**,
  fc_tanh, **fc_proj_out**, **scores**, **prob**,
  prob_left, **prob_right**, **u_neg_div**, **u_pos_div**,
  dstate }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.107.1 Detailed Description

This is MoL Attention Layer Class for Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

11 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

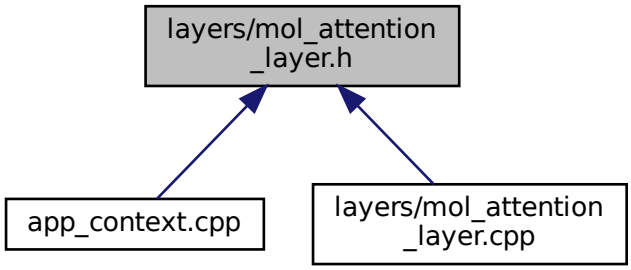
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.108 layers/mol_attention_layer.h File Reference

This is MoL Attention Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.108.1 Detailed Description

This is MoL Attention Layer Class for Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

11 November 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

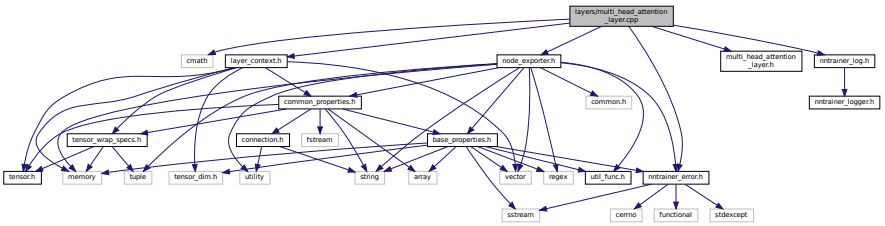
Bug No known bugs except for NYI items

9.109 layers/multi_head_attention_layer.cpp File Reference

This is MultiHeadAttention Layer Class for Neural Network.

```
#include <cmath>
#include <layer_context.h>
#include <multi_head_attention_layer.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
```

Include dependency graph for multi_head_attention_layer.cpp:



Namespaces

- nntrainer

Enumerations

- enum nntrainer::INOUT_INDEX {
  nntrainer::QUERY = 0, **KEY** = 1, **VALUE** = 2, **MASK** = 3,
  nntrainer::OUTPUT = 0, **RETURN_ATTENTION_WEIGHT** = 1 }
- enum nntrainer::AttentionParams {
  query = 0, nntrainer::value = 1, **key** = 2, **weights**,
  query_fc_weight, **query_fc_bias**, **key_fc_weight**, **key_fc_bias**,
  value_fc_weight, **value_fc_bias**, **fc_weight**, **fc_bias**,
  projected_query, **projected_key**, **projected_value**, nntrainer::attention_weight,
  dropout_mask, **attention_output** }

9.109.1 Detailed Description

This is MultiHeadAttention Layer Class for Neural Network.

Copyright (C) 2022 hyeonseok Lee hs89.lee@samsung.com

Date

08 July 2022

See also

https://github.com/nnstreamer/nntrainer https://arxiv.org/abs/1706.↩
03762

Author

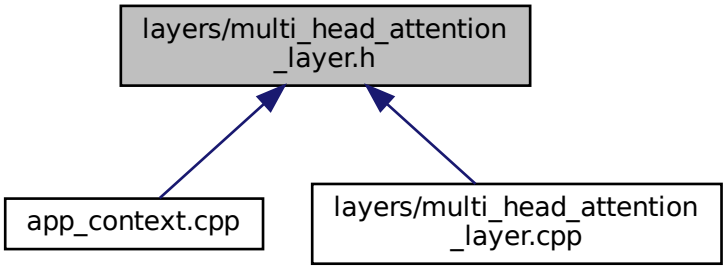
hyeonseok Lee hs89.lee@samsung.com

Bug No known bugs except for NYI items

9.110 layers/multi_head_attention_layer.h File Reference

This is MultiHeadAttention Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.110.1 Detailed Description

This is MultiHeadAttention Layer Class for Neural Network.

Copyright (C) 2022 hyeonseok Lee hs89.lee@samsung.com

Date

08 July 2022

See also

https://github.com/nnstreamer/nntrainer https://arxiv.org/abs/1706.↩
03762

Author

hyeonseok Lee hs89.lee@samsung.com

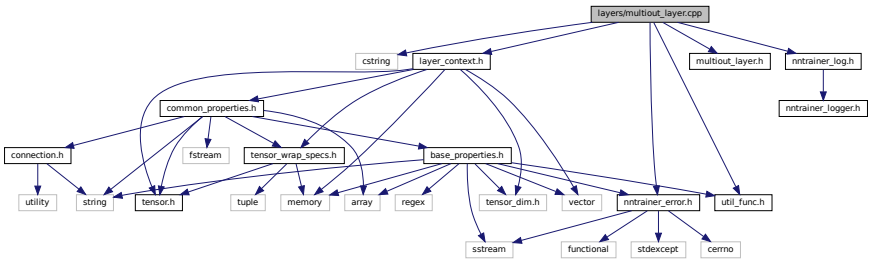
Bug No known bugs except for NYI items

9.111 layers/multiout_layer.cpp File Reference

This is Multi Output Layer Class for Neural Network.

```
#include <cstring>
#include <layer_context.h>
#include <multiout_layer.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <util_func.h>
```

Include dependency graph for multiout_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.111.1 Detailed Description

This is Multi Output Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

05 Nov 2020

See also

https://github.com/nnstreamer/nntrainer

Author

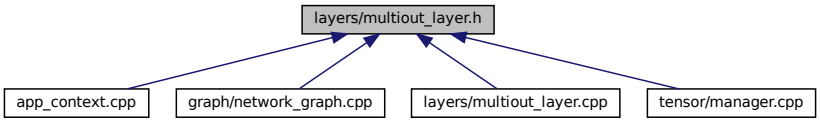
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.112 layers/multiout_layer.h File Reference

This is Multi Output Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.112.1 Detailed Description

This is Multi Output Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

05 Nov 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

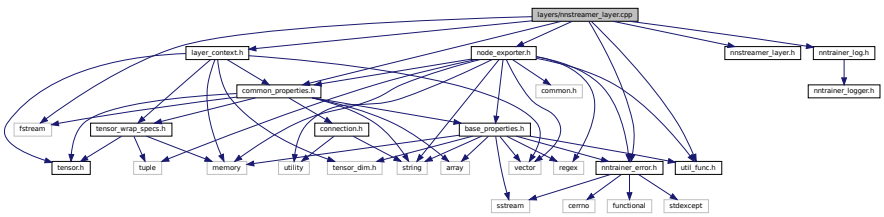
Bug No known bugs except for NYI items

9.113 layers/nnstreamer_layer.cpp File Reference

This is class to encapsulate nnstreamer as a layer of Neural Network.

```
#include <fstream>
#include <common_properties.h>
#include <layer_context.h>
#include <nnstreamer_layer.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for nnstreamer_layer.cpp:



Classes

- class nntrainer::PropsNNSModelPath

  NNSModelPath property.

Namespaces

- nntrainer

Functions

- static int **nntrainer::nnst_info_to_tensor_dim** (ml_tensors_info_h &out_res, TensorDim &dim)

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.113.1 Detailed Description

This is class to encapsulate nnstreamer as a layer of Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

26 October 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

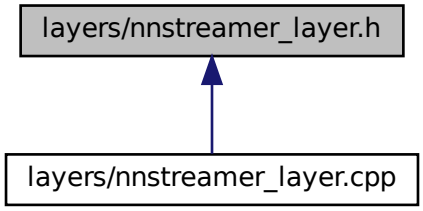
Todo : provide input/output dimensions to nnstreamer for certain frameworks

: support transposing the data to support NCHW nntrainer data to NHWC nnstreamer data

9.114 layers/nnstreamer_layer.h File Reference

This is class to encapsulate nnstreamer as a layer of Neural Network.

This graph shows which files directly or indirectly include this file:



9.114.1 Detailed Description

This is class to encapsulate nnstreamer as a layer of Neural Network.

Date

26 October 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

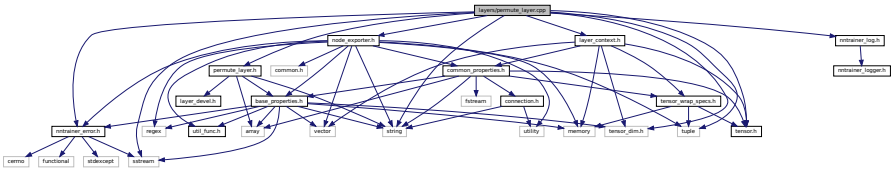
Bug No known bugs except for NYI items

9.115 layers/permute_layer.cpp File Reference

Permute layer to support transpose.

```
#include <sstream>
#include <string>
#include <tuple>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <permute_layer.h>
#include <tensor.h>
#include <tensor_dim.h>
```

Include dependency graph for permute_layer.cpp:



Namespaces

- nntrainer

Functions

- static std::string nntrainer::buildTrasposeString (const std::array< props::PermuteDims, 3 > &arr)

  buildTransposeString based on array

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.115.1 Detailed Description

Permute layer to support transpose.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

06 May 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

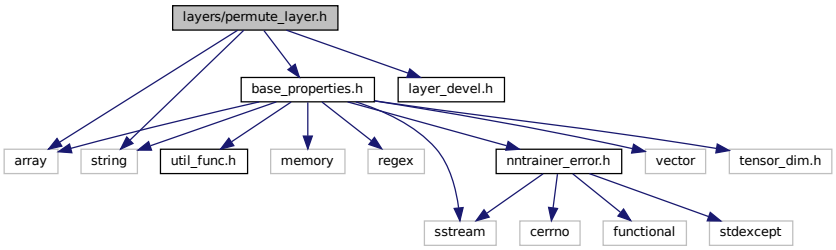
Bug No known bugs except for NYI items

9.116 layers/permute_layer.h File Reference

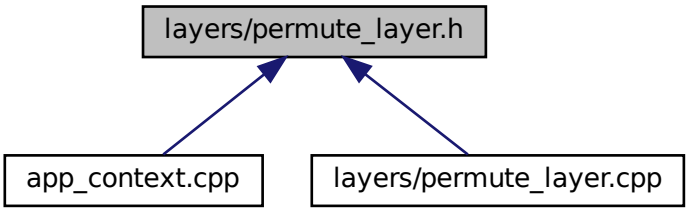
Permute layer to support transpose.

```
#include <array>
#include <string>
#include <base_properties.h>
#include <layer_devel.h>
```

Include dependency graph for permute_layer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::props::PermuteDims

  PermuteDims property, direction property describes the axis to be transposed. to be used with array.
- class nntrainer::PermuteLayer

  Permute layer to transpose a tensor.

Namespaces

- nntrainer

9.116.1 Detailed Description

Permute layer to support transpose.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

06 May 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

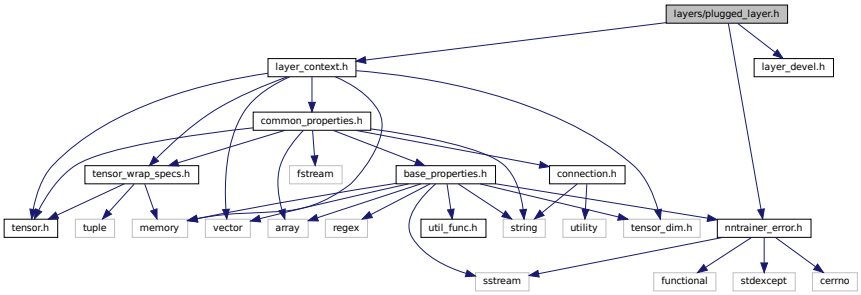
Bug No known bugs except for NYI items

9.117 layers/plugged_layer.h File Reference

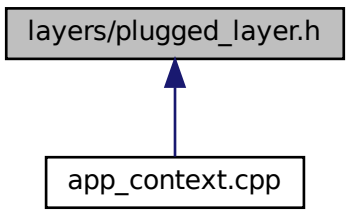
This file contains a wrapper for a plugged layer, INTERNAL USE ONLY.

```
#include <layer_context.h>
#include <layer_devel.h>
#include <nntrainer_error.h>
```

Include dependency graph for plugged_layer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::internal::PluggedLayer

  PluggedLayer to wrap a layer from shared object file.

Namespaces

- nntrainer

9.117.1 Detailed Description

This file contains a wrapper for a plugged layer, INTERNAL USE ONLY.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

27 January 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

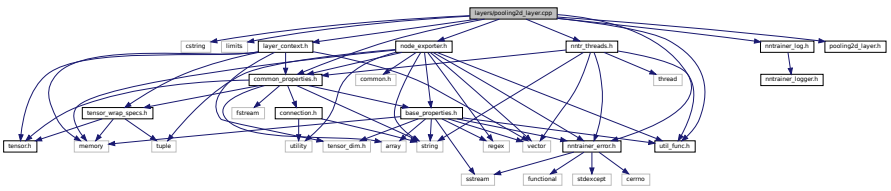
Bug No known bugs except for NYI items

9.118 layers/pooling2d_layer.cpp File Reference

This is 2 Dimensional Pooling Layer Class for Neural Network.

```
#include <cstring>
#include <limits>
#include <common_properties.h>
#include <layer_context.h>
#include <nntr_threads.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <pooling2d_layer.h>
#include <util_func.h>
```

Include dependency graph for pooling2d_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.118.1 Detailed Description

This is 2 Dimensional Pooling Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

12 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

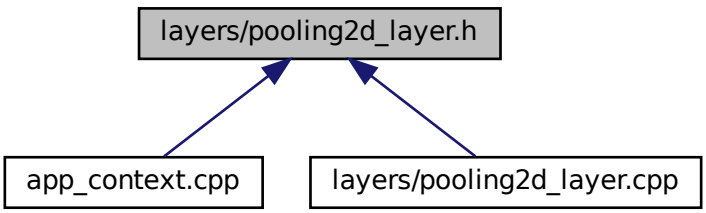
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.119 layers/pooling2d_layer.h File Reference

This is 2 Dimensional Pooling Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.119.1 Detailed Description

This is 2 Dimensional Pooling Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

12 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

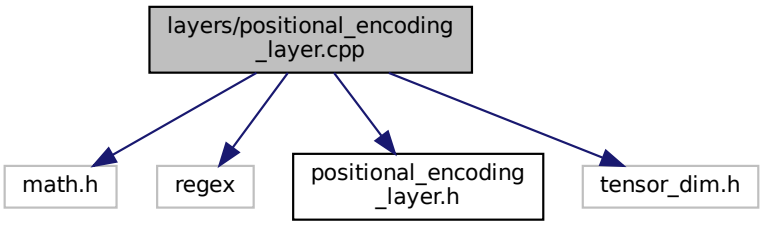
Bug No known bugs except for NYI items

9.120 layers/positional_encoding_layer.cpp File Reference

This file contains the positional encoding layer in transformer.

```
#include <math.h>
#include <regex>
#include <positional_encoding_layer.h>
#include <tensor_dim.h>
```

Include dependency graph for positional_encoding_layer.cpp:



Namespaces

- nntrainer

Enumerations

- enum **PositionalEncodingParams** { **positional_encoding** }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.120.1 Detailed Description

This file contains the positional encoding layer in transformer.

Date

16 August 2022

See also

https://github.com/nnstreamer/nntrainer https://arxiv.org/abs/1607.↩
06450

Author

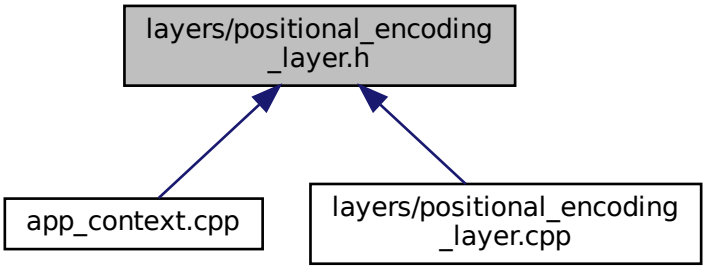
Hyeonseok Lee hs89.lee@samsung.com

Bug No known bugs except for NYI items

9.121 layers/positional_encoding_layer.h File Reference

This file contains the positional encoding layer in transformer.

This graph shows which files directly or indirectly include this file:



9.121.1 Detailed Description

This file contains the positional encoding layer in transformer.

Copyright (C) 2022 Hyeonseok Lee hs89.lee@samsung.com

Date

16 August 2022

See also

https://github.com/nnstreamer/nntrainer https://arxiv.org/abs/1607.↩
06450

Author

Hyeonseok Lee hs89.lee@samsung.com

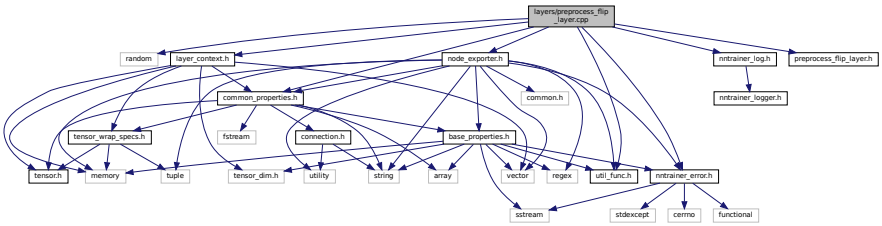
Bug No known bugs except for NYI items

9.122 layers/preprocess_flip_layer.cpp File Reference

This is Preprocess Random Flip Layer Class for Neural Network.

```
#include <random>
#include <common_properties.h>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <preprocess_flip_layer.h>
#include <util_func.h>
```

Include dependency graph for preprocess_flip_layer.cpp:



Namespaces

- nntrainer

9.122.1 Detailed Description

This is Preprocess Random Flip Layer Class for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

20 January 2020

See also

https://github.com/nnstreamer/nntrainer

Author

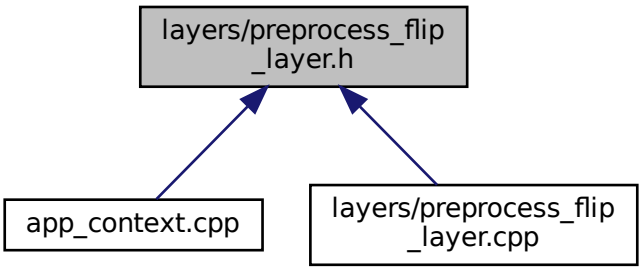
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.123 layers/preprocess_flip_layer.h File Reference

This is Preprocess Random Flip Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.123.1 Detailed Description

This is Preprocess Random Flip Layer Class for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

20 January 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

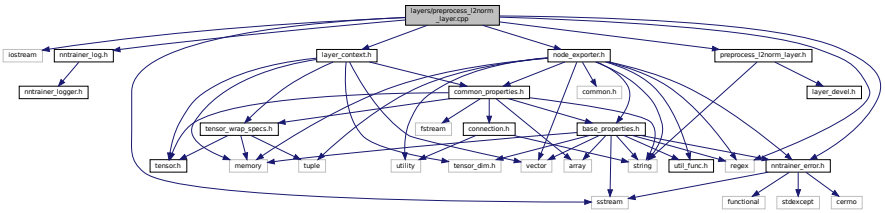
Bug No known bugs except for NYI items

9.124 layers/preprocess_l2norm_layer.cpp File Reference

This file contains the simple l2norm layer which normalizes the given feature.

```
#include <iostream>
#include <regex>
#include <sstream>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <preprocess_l2norm_layer.h>
```

Include dependency graph for preprocess_l2norm_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.124.1 Detailed Description

This file contains the simple l2norm layer which normalizes the given feature.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 Jan 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

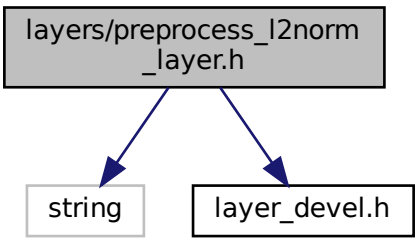
Bug No known bugs except for NYI items

9.125 layers/preprocess_l2norm_layer.h File Reference

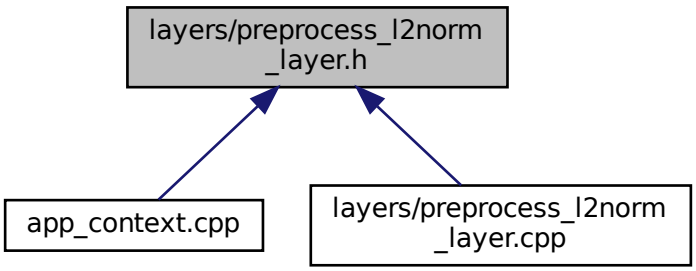
This file contains the simple l2norm layer which normalizes the given feature.

```
#include <string>
#include <layer_devel.h>
```

Include dependency graph for preprocess_l2norm_layer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::PreprocessL2NormLayer

  Layer class that l2normalizes a feature vector.

Namespaces

- nntrainer

9.125.1 Detailed Description

This file contains the simple l2norm layer which normalizes the given feature.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 Jan 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

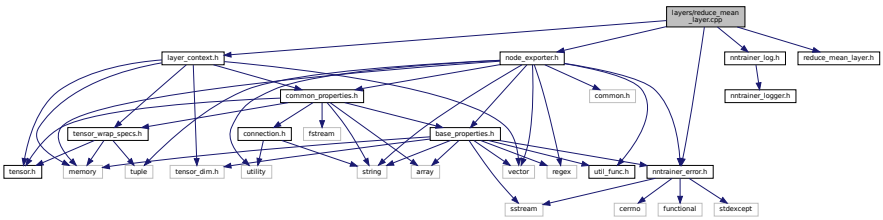
Bug No known bugs except for NYI items

9.126 layers/reduce_mean_layer.cpp File Reference

This is Reduce Mean Layer Class for Neural Network.

```
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <reduce_mean_layer.h>
```

Include dependency graph for reduce_mean_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.126.1 Detailed Description

This is Reduce Mean Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

25 Nov 2021

See also

https://github.com/nnstreamer/nntrainer

Author

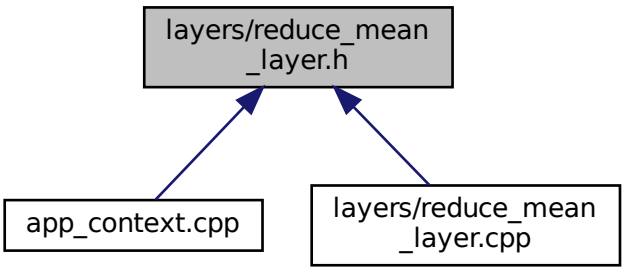
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.127 layers/reduce_mean_layer.h File Reference

This is Reduce Mean Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.127.1 Detailed Description

This is Reduce Mean Layer Class for Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

25 Nov 2021

See also

https://github.com/nnstreamer/nntrainer

Author

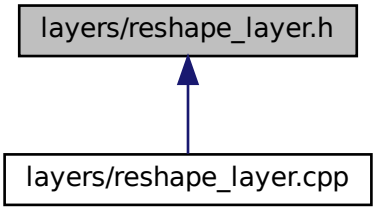
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.128 layers/reshape_layer.h File Reference

This is Reshape Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.128.1 Detailed Description

This is Reshape Layer Class for Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

16 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

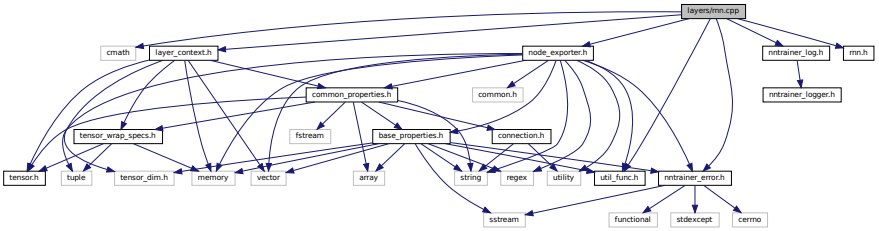
Bug No known bugs except for NYI items

9.129 layers/rnn.cpp File Reference

This is Recurrent Layer Class of Neural Network.

```
#include <cmath>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <rnn.h>
#include <util_func.h>
```

Include dependency graph for rnn.cpp:



Namespaces

- nntrainer

Enumerations

- enum **RNNParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **hidden_state**, **dropout_mask** }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.129.1 Detailed Description

This is Recurrent Layer Class of Neural Network.

Copyright (C) 2021 Jijoong Moon jijoong.moon@samsung.com

Date

17 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

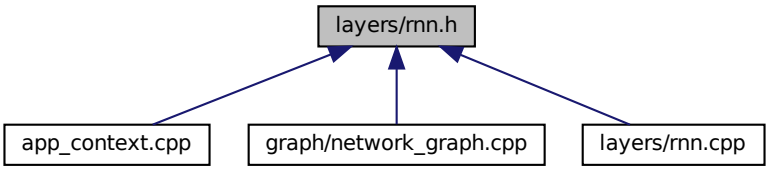
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.130 layers/rnn.h File Reference

This is Recurrent Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.130.1 Detailed Description

This is Recurrent Layer Class of Neural Network.

Copyright (C) 2021 Jijoong Moon jijoong.moon@samsung.com

Date

17 March 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

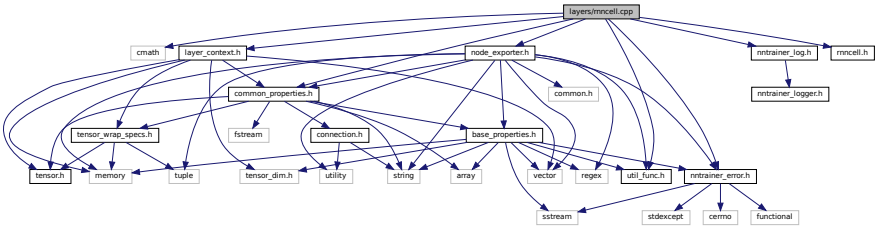
9.131 layers/rnncell.cpp File Reference

This is Recurrent Cell Layer Class of Neural Network.

```
#include <cmath>
#include <common_properties.h>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <rnncell.h>
```

```
#include <util_func.h>
```

Include dependency graph for rnncell.cpp:



Namespaces

- nntrainer

Enumerations

- enum **RNNCellParams** { **weight_ih**, **weight_hh**, **bias_h**, **bias_ih**, **bias_hh**, **dropout_mask** }

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.131.1 Detailed Description

This is Recurrent Cell Layer Class of Neural Network.

Copyright (C) 2021 hyeonseok lee hs89.lee@samsung.com

Date

29 Oct 2021

See also

https://github.com/nnstreamer/nntrainer

Author

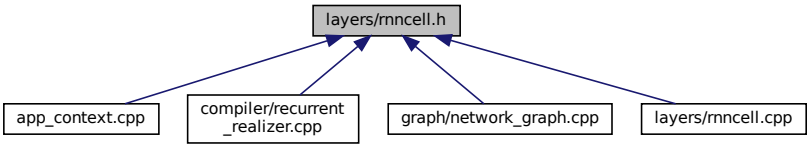
hyeonseok lee hs89.lee@samsung.com

Bug No known bugs except for NYI items

9.132 layers/rnncell.h File Reference

This is Recurrent Layer Cell Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.132.1 Detailed Description

This is Recurrent Layer Cell Class of Neural Network.

Copyright (C) 2021 hyeonseok lee hs89.lee@samsung.com

Date

29 Oct 2021

See also

https://github.com/nnstreamer/nntrainer

Author

hyeonseok lee hs89.lee@samsung.com

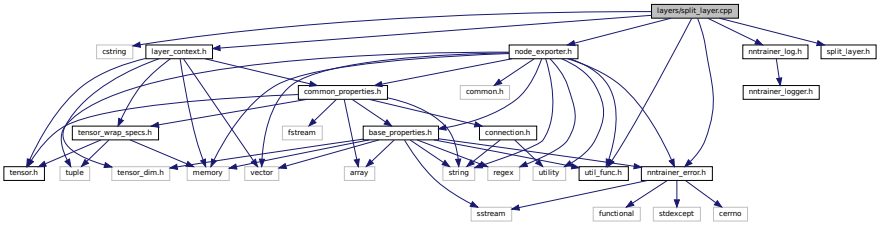
Bug No known bugs except for NYI items

9.133 layers/split_layer.cpp File Reference

This is Split Layer Class for Neural Network.

```
#include <cstring>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <split_layer.h>
#include <util_func.h>
```

Include dependency graph for split_layer.cpp:



Namespaces

- nntrainer

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.133.1 Detailed Description

This is Split Layer Class for Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

21 May 2021

See also

https://github.com/nnstreamer/nntrainer

Author

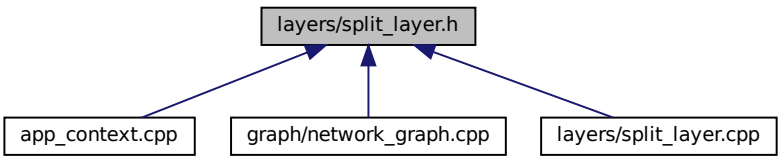
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.134 layers/split_layer.h File Reference

This is Split Layer Class for Neural Network.

This graph shows which files directly or indirectly include this file:



9.134.1 Detailed Description

This is Split Layer Class for Neural Network.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

21 May 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

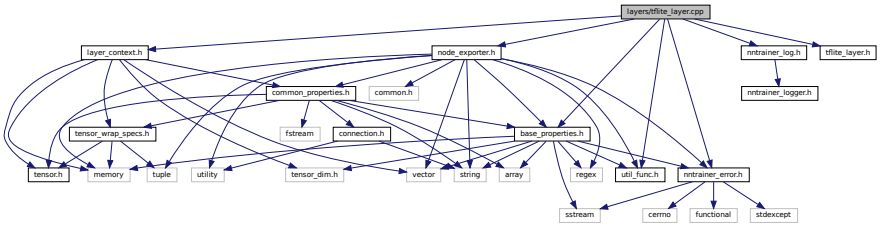
Todo Add support for uneven splits. For now, this can be acheived with combination of split and concat layers.

9.135 layers/tflite_layer.cpp File Reference

This is class to encapsulate tflite as a layer of Neural Network.

```
#include <base_properties.h>
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <tflite_layer.h>
#include <util_func.h>
```

Include dependency graph for tflite_layer.cpp:



Classes

- class nntrainer::PropsTflModelPath

  TflModelPath property.

Namespaces

- nntrainer

9.135.1 Detailed Description

This is class to encapsulate tflite as a layer of Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

26 October 2020

See also

https://github.com/nnstreamer/nntrainer

Author

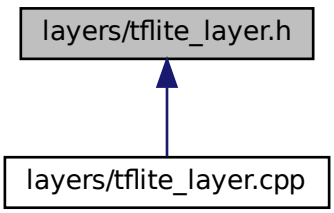
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.136 layers/tflite_layer.h File Reference

This is class to encapsulate tflite as a layer of Neural Network.

This graph shows which files directly or indirectly include this file:



9.136.1 Detailed Description

This is class to encapsulate tflite as a layer of Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

3 November 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

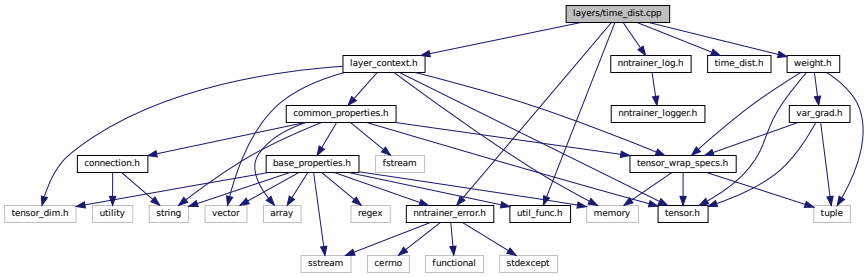
Bug No known bugs except for NYI items

9.137 layers/time_dist.cpp File Reference

This is Time Distributed Layer Class of Neural Network.

```
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <time_dist.h>
#include <util_func.h>
#include <weight.h>
```

Include dependency graph for time_dist.cpp:



Namespaces

- nntrainer

Functions

- static void **nntrainer::reshape** (Tensor &m)

Variables

- static constexpr size_t **nntrainer::SINGLE_INOUT_IDX** = 0

9.137.1 Detailed Description

This is Time Distributed Layer Class of Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

01 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

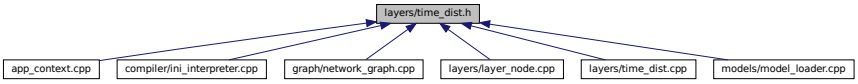
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.138 layers/time_dist.h File Reference

This is Time Distributed Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.138.1 Detailed Description

This is Time Distributed Layer Class of Neural Network.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

01 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

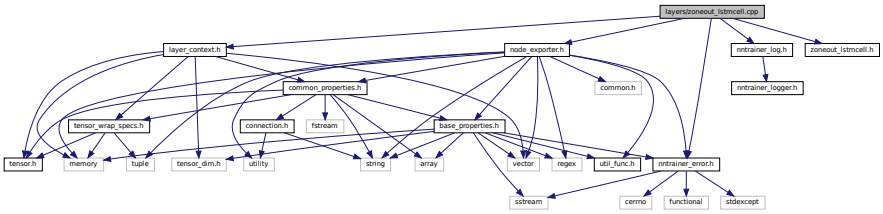
Bug No known bugs except for NYI items

9.139 layers/zoneout_lstmcell.cpp File Reference

This is ZoneoutLSTMCell Layer Class of Neural Network.

```
#include <layer_context.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <zoneout_lstmcell.h>
```

Include dependency graph for zoneout_lstmcell.cpp:



Namespaces

- nntrainer

Enumerations

- enum **ZoneoutLSTMParams** {
  weight_ih, **weight_hh**, **bias_h**, **bias_ih**,
  bias_hh, **ifgo**, **hidden_state_zoneout_mask**, **cell_state_zoneout_mask**,
  lstm_cell_state }

9.139.1 Detailed Description

This is ZoneoutLSTMCell Layer Class of Neural Network.

Copyright (C) 2021 hyeonseok lee hs89.lee@samsung.com

Date

30 November 2021

See also

https://github.com/nnstreamer/nntrainer https://arxiv.org/pdf/1606.↩
01305.pdf https://github.com/teganmaharaj/zoneout

Author

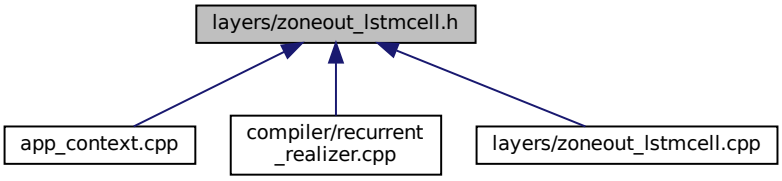
hyeonseok lee hs89.lee@samsung.com

Bug No known bugs except for NYI items

9.140 layers/zoneout_lstmcell.h File Reference

This is ZoneoutLSTMCell Layer Class of Neural Network.

This graph shows which files directly or indirectly include this file:



9.140.1 Detailed Description

This is ZoneoutLSTMCell Layer Class of Neural Network.

Copyright (C) 2021 hyeonseok lee hs89.lee@samsung.com

Date

30 November 2021

See also

https://github.com/nnstreamer/nntrainer https://arxiv.org/pdf/1606.↩
01305.pdf https://github.com/teganmaharaj/zoneout

Author

hyeonseok lee hs89.lee@samsung.com

Bug No known bugs except for NYI items

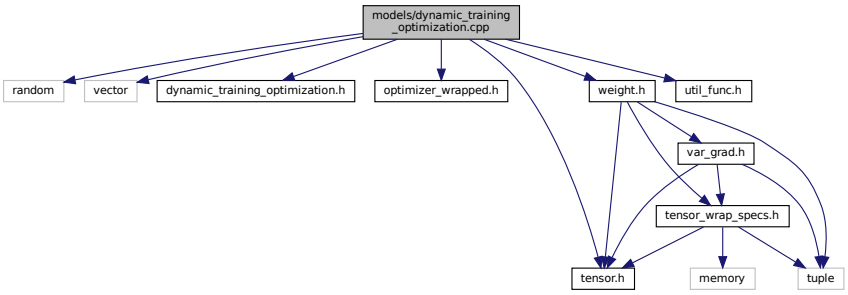
9.141 models/dynamic_training_optimization.cpp File Reference

This is Dynamic Training Optimization for Neural Network.

```
#include <random>
#include <vector>
#include <dynamic_training_optimization.h>
#include <optimizer_wrapped.h>
#include <tensor.h>
#include <util_func.h>
```

```
#include <weight.h>
```

Include dependency graph for dynamic_training_optimization.cpp:



Namespaces

- nntrainer

9.141.1 Detailed Description

This is Dynamic Training Optimization for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

5 January 2021

See also

https://github.com/nnstreamer/nntrainer

Author

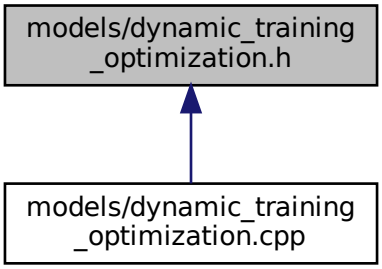
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.142 models/dynamic_training_optimization.h File Reference

This is Dynamic Training Optimization for Neural Network.

This graph shows which files directly or indirectly include this file:



9.142.1 Detailed Description

This is Dynamic Training Optimization for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

4 January 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Dynamic training aims to optimize the cost of applying the gradient. The cost of applying the gradient includes the cost of the optimizer (adam, etc) where the optimizer variables are updated, and the cost of actually updating the weights (which can be non-trivial with bigger weights and distributed training).

There are two supported modes:

1. Gradient Mode: The already calculated gradient is used to estimate if this gradient must be used to update the weight, or if this update must be skipped.
2. Derivative Mode: This mode tries to estimate an approximate gradient with low cost in order to save the cost of calculating gradient. This cost of calculating gradient is wasted if the gradient is not going to be applied.

There are two supported reduction operations which reduce the gradient and the weight to a single value in order to compare it with a threshold. If the reduced value is less than threshold, the update is performed with some probabilty proportional to the value. If the reduced value is higher than threshold, then the update is always performed.

9.143 models/execution_mode.h File Reference

This is mode of executions.

Namespaces

- nntrainer

Enumerations

- enum nntrainer::ExecutionMode { **TRAIN**, nntrainer::ExecutionMode::INFERENCE, nntrainer::ExecutionMode::VALIDATE }

  class telling the execution mode of the model/operation

9.143.1 Detailed Description

This is mode of executions.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

25 June 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

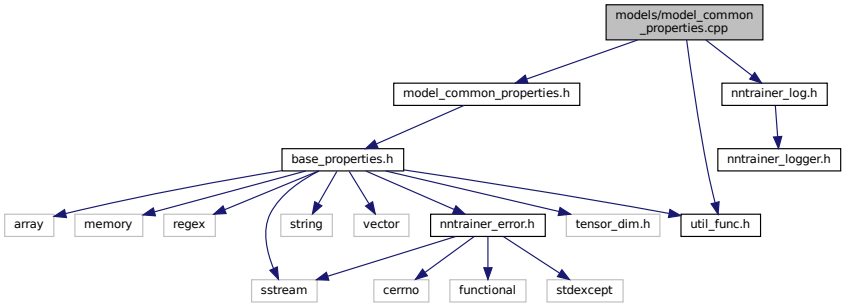
Bug No known bugs except for NYI items

9.144 models/model_common_properties.cpp File Reference

This file contains common properties for model.

```
#include <model_common_properties.h>
#include <nntrainer_log.h>
#include <util_func.h>
```

Include dependency graph for model_common_properties.cpp:



Namespaces

- nntrainer

9.144.1 Detailed Description

This file contains common properties for model.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

27 Aug 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

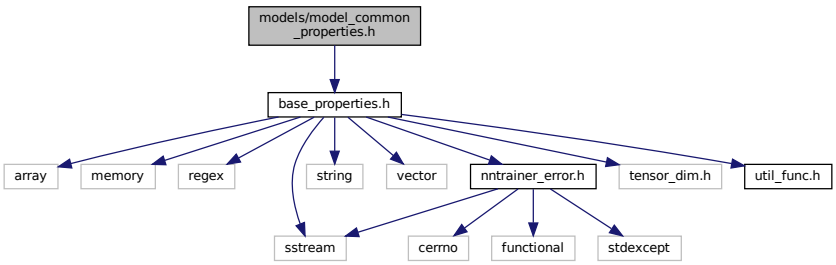
Bug No known bugs except for NYI items

9.145 models/model_common_properties.h File Reference

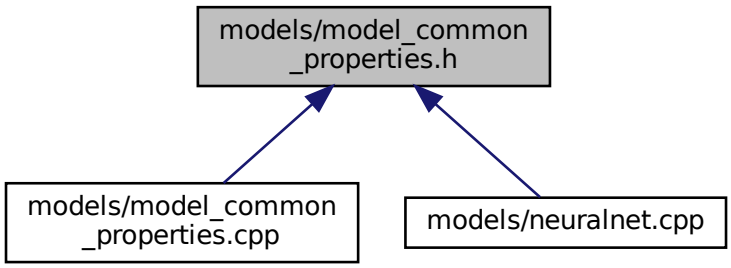
This file contains common properties for model.

```
#include <base_properties.h>
```

Include dependency graph for model_common_properties.h:



This graph shows which files directly or indirectly include this file:



9.145.1 Detailed Description

This file contains common properties for model.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

27 Aug 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

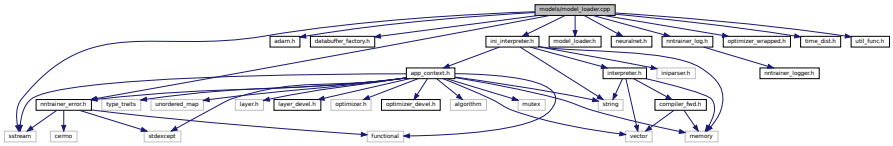
9.146 models/model_loader.cpp File Reference

This is model loader class for the Neural Network.

```
#include <sstream>
#include <adam.h>
#include <databuffer_factory.h>
#include <ini_interpreter.h>
#include <model_loader.h>
#include <neuralnet.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <optimizer_wrapped.h>
```

```
#include <time_dist.h>
#include <util_func.h>
```

Include dependency graph for model_loader.cpp:



Namespaces

- nntrainer

Macros

- #define **NN_INI_RETURN_STATUS**()

9.146.1 Detailed Description

This is model loader class for the Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

5 August 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.146.2 Macro Definition Documentation

9.146.2.1 NN_INI_RETURN_STATUS

```
#define NN_INI_RETURN_STATUS( )
```

Value:

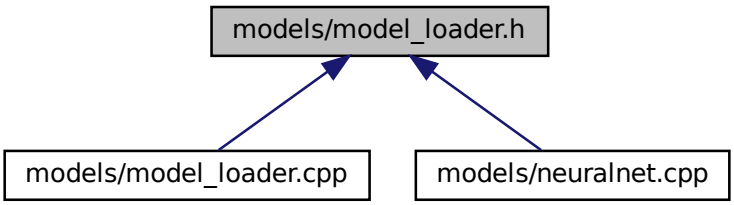
```
  do {                                \
    if (status != ML_ERROR_NONE) {    \
      iniparser_freedict(ini);        \
      return status;                  \
    }                                 \
  } while (0)
```

Definition at line 35 of file model_loader.cpp.

9.147 models/model_loader.h File Reference

This is model loader class for the Neural Network.

This graph shows which files directly or indirectly include this file:



9.147.1 Detailed Description

This is model loader class for the Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

5 August 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

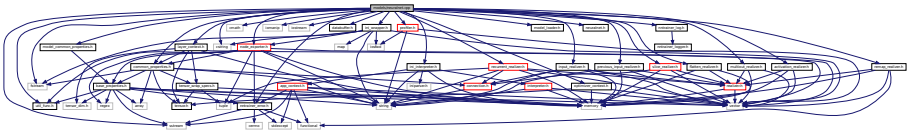
Bug No known bugs except for NYI items

9.148 models/neuralnet.cpp File Reference

This is Neural Network Class.

```
#include "layer_context.h"
#include "model_common_properties.h"
#include <cmath>
#include <cstring>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <activation_realizer.h>
#include <common_properties.h>
#include <databuffer.h>
#include <flatten_realizer.h>
#include <ini_interpreter.h>
#include <ini_wrapper.h>
#include <input_realizer.h>
#include <model_loader.h>
#include <multiout_realizer.h>
#include <neuralnet.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <optimizer_context.h>
#include <previous_input_realizer.h>
#include <profiler.h>
#include <recurrent_realizer.h>
#include <remap_realizer.h>
#include <slice_realizer.h>
#include <util_func.h>
```

Include dependency graph for neuralnet.cpp:



Namespaces

- nntrainer

Macros

- #define ML_TRAIN_SUMMARY_MODEL_TRAIN_LOSS 101

  Internal enum values for nntrainer to summarize model accuracy & loss.
- #define **ML_TRAIN_SUMMARY_MODEL_VALID_LOSS** 102
- #define **ML_TRAIN_SUMMARY_MODEL_VALID_ACCURACY** 103

Functions

- void **nntrainer::swap** (NeuralNetwork &lhs, NeuralNetwork &rhs)

9.148.1 Detailed Description

This is Neural Network Class.

Date

04 December 2019

See also

https://github.com/nnstreamer/nntrainer

Author

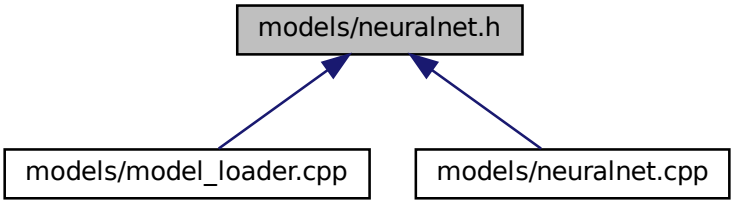
Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.149 models/neuralnet.h File Reference

This is Neural Network Class.

This graph shows which files directly or indirectly include this file:



9.149.1 Detailed Description

This is Neural Network Class.

Date

04 December 2019

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

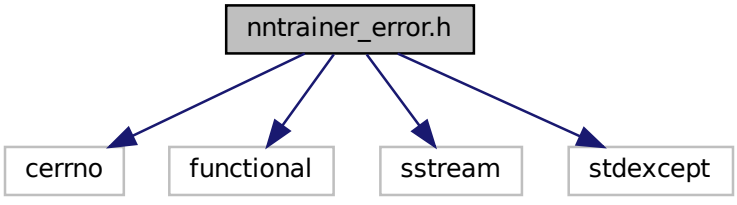
Bug No known bugs except for NYI items

9.150 nntrainer_error.h File Reference

NNTrainer Error Codes.

```
#include <cerrno>
#include <functional>
#include <sstream>
#include <stdexcept>
```

Include dependency graph for nntrainer_error.h:



Classes

- class nntrainer::exception::ErrorNotification< Err, >

  Error Notification class, error is thrown when the class is destroyed. DO NOT use this outside as this contains throwing destructor.
- struct nntrainer::exception::not_supported

  derived class of invalid argument to represent specific functionality not supported
- struct nntrainer::exception::permission_denied

  derived class of invalid argument to represent permission is denied

Namespaces

- nntrainer
- nntrainer::exception

Macros

- #define **ML_ERROR_BAD_ADDRESS** (-EFAULT)
- #define **ML_ERROR_RESULT_OUT_OF_RANGE** (-ERANGE)
- #define **NNTR_THROW_IF**(pred, err)
- #define **NNTR_THROW_IF_CLEANUP**(pred, err, cleanup_func)
- #define ESTRPIPE EPIPE
- #define **_ERROR_UNKNOWN** (-1073741824LL)
- #define **TIZEN_ERROR_TIMED_OUT** (TIZEN_ERROR_UNKNOWN + 1)
- #define **TIZEN_ERROR_NOT_SUPPORTED** (TIZEN_ERROR_UNKNOWN + 2)
- #define **TIZEN_ERROR_PERMISSION_DENIED** (-EACCES)
- #define **TIZEN_ERROR_OUT_OF_MEMORY** (-ENOMEM)

Enumerations

- enum ml_error_e { ML_ERROR_NONE = 0, ML_ERROR_INVALID_PARAMETER = -EINVAL, ML_ERROR_TRY_AGAIN, ML_ERROR_UNKNOWN = _ERROR_UNKNOWN, ML_ERROR_TIMED_OUT = (_ERROR_UNKNOWN + 1), ML_ERROR_NOT_SUPPORTED, ML_ERROR_PERMISSION_DEN = -EACCES, ML_ERROR_OUT_OF_MEMORY = -ENOMEM }

9.150.1 Detailed Description

NNTrainer Error Codes.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

03 April 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.150.2 Macro Definition Documentation

9.150.2.1 ESTRPIPE

```
#define ESTRPIPE EPIPE
```

https://gitlab.freedesktop.org/dude/gst-plugins-base/commit/89095e7f91cfbfe625ec2522da4

Definition at line 46 of file nntrainer_error.h.

9.150.2.2 NNTR_THROW_IF

```
#define NNTR_THROW_IF(
              pred,
              err )
```

Value:

```
  if ((pred))                    \
    nntrainer::exception::ErrorNotification<err> {}
```

Definition at line 30 of file nntrainer_error.h.

9.150.2.3 NNTR_THROW_IF_CLEANUP

```
#define NNTR_THROW_IF_CLEANUP(
              pred,
              err,
              cleanup_func )
```

Value:

```
  if ((pred))                                          \
    nntrainer::exception::ErrorNotification<err> { cleanup_func }
```

Definition at line 34 of file nntrainer_error.h.

9.150.3 Enumeration Type Documentation

9.150.3.1 ml_error_e

```
enum ml_error_e
```

Enumerator

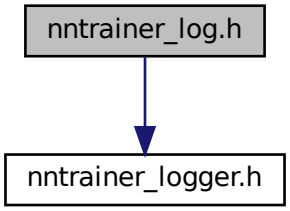
ML_ERROR_NONE	Success!
ML_ERROR_INVALID_PARAMETER	Invalid parameter
ML_ERROR_TRY_AGAIN	The pipeline is not ready, yet (not negotiated, yet)
ML_ERROR_UNKNOWN	Unknown error
ML_ERROR_TIMED_OUT	Time out
ML_ERROR_NOT_SUPPORTED	The feature is not supported
ML_ERROR_PERMISSION_DENIED	Permission denied
ML_ERROR_OUT_OF_MEMORY	Out of memory (Since 6.0)

Definition at line 54 of file nntrainer_error.h.

9.151 nntrainer_log.h File Reference

NNTrainer Logger. Log Util for NNTrainer.

```
#include <nntrainer_logger.h>
```
Include dependency graph for nntrainer_log.h:



Macros

- #define **TAG_NAME** "nntrainer"
- #define **ml_logi**(format, args...)
- #define ml_logw(format, args...)

  args)
- #define ml_loge(format, args...)

  args)
- #define ml_logd(format, args...)

  args)

9.151.1 Detailed Description

NNTrainer Logger. Log Util for NNTrainer.

Date

06 April 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.151.2 Macro Definition Documentation

9.151.2.1 ml_logd

```
#define ml_logd(
              format,
              args...  )
```

Value:

```
__nntrainer_log_print(NNTRAINER_LOG_DEBUG, "(%s:%s:%d) " format, __FILE__, \
                      __func__, __LINE__,
```

args)

Definition at line 77 of file nntrainer_log.h.

9.151.2.2 ml_loge

```
#define ml_loge(
              format,
              args...  )
```

Value:

```
  __nntrainer_log_print(NNTRAINER_LOG_ERROR, "(%s:%s:%d) " format, __FILE__, \
                        __func__, __LINE__,
```

args)

Definition at line 71 of file nntrainer_log.h.

9.151.2.3 ml_logi

```
#define ml_logi(
              format,
              args...  )
```

Value:

```
  __nntrainer_log_print(NNTRAINER_LOG_INFO, "(%s:%s:%d) " format, __FILE__, \
                        __func__, __LINE__,
```

Definition at line 59 of file nntrainer_log.h.

9.151.2.4 ml_logw

```
#define ml_logw(
              format,
              args...  )
```

Value:

```
  __nntrainer_log_print(NNTRAINER_LOG_WARN, "(%s:%s:%d) " format, __FILE__, \
                        __func__, __LINE__,
```

args)

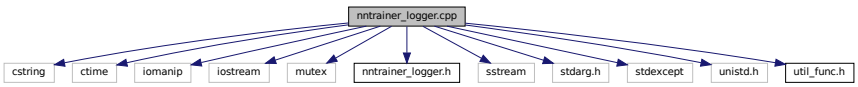
Definition at line 65 of file nntrainer_log.h.

9.152 nntrainer_logger.cpp File Reference

NNTrainer Logger This allows to logging nntrainer logs.

```
#include <cstring>
#include <ctime>
#include <iomanip>
#include <iostream>
#include <mutex>
#include <nntrainer_logger.h>
#include <sstream>
#include <stdarg.h>
#include <stdexcept>
#include <unistd.h>
#include <util_func.h>
```

Include dependency graph for nntrainer_logger.cpp:



Namespaces

- nntrainer

Functions

- void nntrainer::__nntrainer_log_print (nntrainer_loglevel loglevel, const std::string format_str,...)

  Interface function for C.

9.152.1 Detailed Description

NNTrainer Logger This allows to logging nntrainer logs.

Date

02 April 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.153 nntrainer_logger.h File Reference

NNTrainer Logger This allows to logging nntrainer logs.

This graph shows which files directly or indirectly include this file:

Functions

- void __nntrainer_log_print (nntrainer_loglevel loglevel, const std::string format,...)

  Interface function for C.

9.153.1 Detailed Description

NNTrainer Logger This allows to logging nntrainer logs.

Date

02 April 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

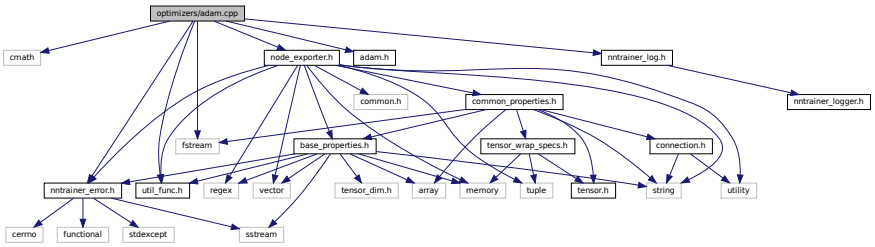
Bug No known bugs except for NYI items

9.154 optimizers/adam.cpp File Reference

This is the Adam optimizer.

```
#include <cmath>
#include <fstream>
#include <adam.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <util_func.h>
```

Include dependency graph for adam.cpp:



Namespaces

- nntrainer

Enumerations

- enum **AdamParams** { **wm**, **wv** }

9.154.1 Detailed Description

This is the Adam optimizer.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

6 October 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

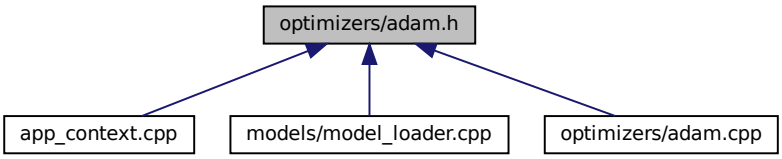
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.155 optimizers/adam.h File Reference

This is the Adam optimizer.

This graph shows which files directly or indirectly include this file:



9.155.1 Detailed Description

This is the Adam optimizer.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

6 October 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.156 optimizers/lr_scheduler.h File Reference

This is Learning Rate Scheduler interface class.

9.156.1 Detailed Description

This is Learning Rate Scheduler interface class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

09 December 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

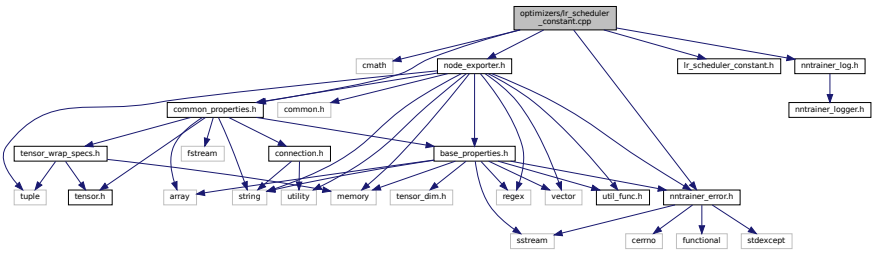
Bug No known bugs except for NYI items

9.157 optimizers/lr_scheduler_constant.cpp File Reference

This is Constant Learning Rate Scheduler class.

```
#include <cmath>
#include <common_properties.h>
#include <lr_scheduler_constant.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
```

Include dependency graph for lr_scheduler_constant.cpp:



Namespaces

- nntrainer

9.157.1 Detailed Description

This is Constant Learning Rate Scheduler class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

09 December 2021

See also

https://github.com/nnstreamer/nntrainer

Author

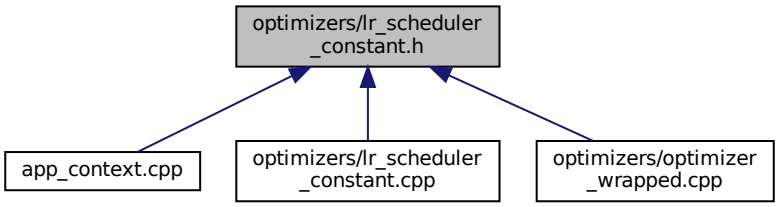
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.158 optimizers/lr_scheduler_constant.h File Reference

This is Constant Learning Rate Scheduler class.

This graph shows which files directly or indirectly include this file:



9.158.1 Detailed Description

This is Constant Learning Rate Scheduler class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

09 December 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

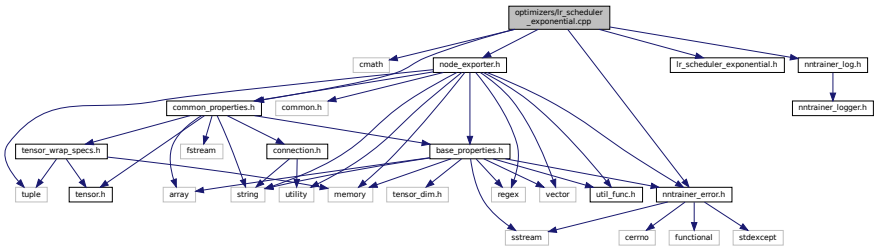
Bug No known bugs except for NYI items

9.159 optimizers/lr_scheduler_exponential.cpp File Reference

This is Exponential Learning Rate Scheduler class.

```
#include <cmath>
#include <common_properties.h>
#include <lr_scheduler_exponential.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
```

Include dependency graph for lr_scheduler_exponential.cpp:



Namespaces

- nntrainer

9.159.1 Detailed Description

This is Exponential Learning Rate Scheduler class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

09 December 2021

See also

https://github.com/nnstreamer/nntrainer

Author

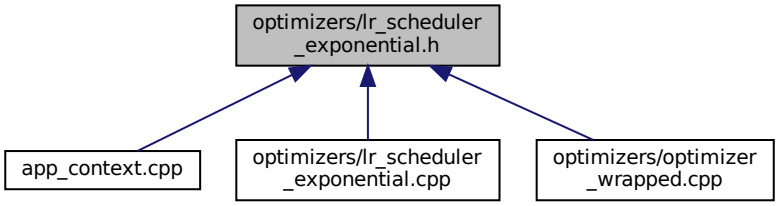
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.160 optimizers/lr_scheduler_exponential.h File Reference

This is Exponential Learning Rate Scheduler class.

This graph shows which files directly or indirectly include this file:



9.160.1 Detailed Description

This is Exponential Learning Rate Scheduler class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

09 December 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

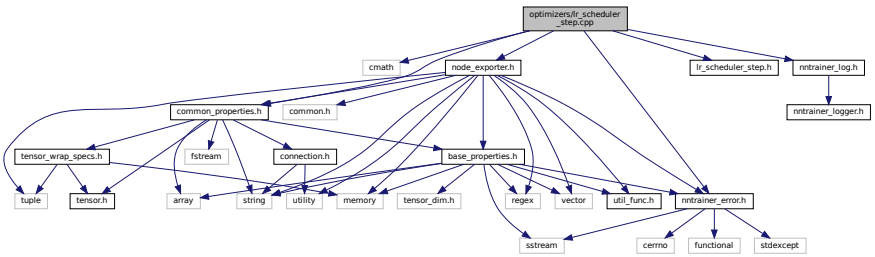
9.161 optimizers/lr_scheduler_step.cpp File Reference

This is Step Learning Rate Scheduler class.

```
#include <cmath>
#include <common_properties.h>
#include <lr_scheduler_step.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
```

```
#include <node_exporter.h>
```

Include dependency graph for lr_scheduler_step.cpp:



Namespaces

- nntrainer

9.161.1 Detailed Description

This is Step Learning Rate Scheduler class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

09 December 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

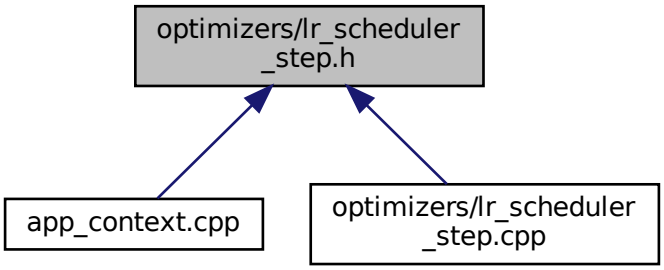
Bug No known bugs except for NYI items

given the iteration range [it_0, it_1, ... it_n], learning rates [lr_0, lr_1 ... lr_n_1], and iteration iter, find index i such that it_i_1 < iter <= it_i and return lr_i_1.

9.162 optimizers/lr_scheduler_step.h File Reference

This is Step Learning Rate Scheduler class.

This graph shows which files directly or indirectly include this file:



9.162.1 Detailed Description

This is Step Learning Rate Scheduler class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

13 December 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

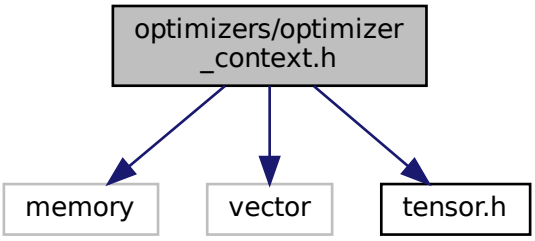
Bug No known bugs except for NYI items

9.163 optimizers/optimizer_context.h File Reference

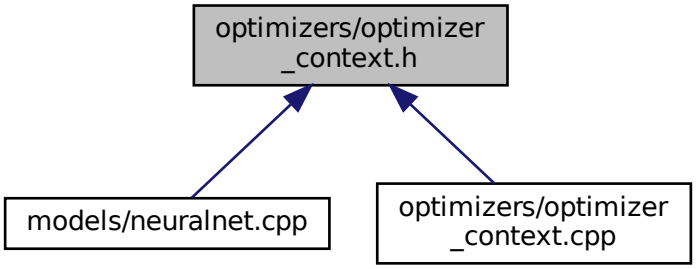
This is the layer context for each layer.

```
#include <memory>
#include <vector>
#include <tensor.h>
```

Include dependency graph for optimizer_context.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::RunOptimizerContext

Namespaces

- nntrainer

9.163.1 Detailed Description

This is the layer context for each layer.

This is the optimizer context for each optimizer.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

30 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

30 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

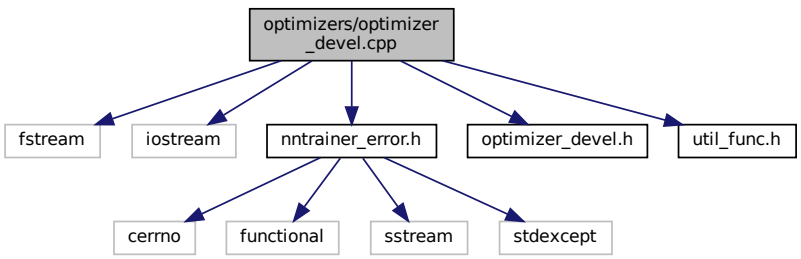
Bug No known bugs except for NYI items

9.164 optimizers/optimizer_devel.cpp File Reference

This is Optimizer internal interface class.

```
#include <fstream>
#include <iostream>
#include <nntrainer_error.h>
#include <optimizer_devel.h>
#include <util_func.h>
```

Include dependency graph for optimizer_devel.cpp:



Namespaces

- nntrainer

9.164.1 Detailed Description

This is Optimizer internal interface class.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

08 April 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

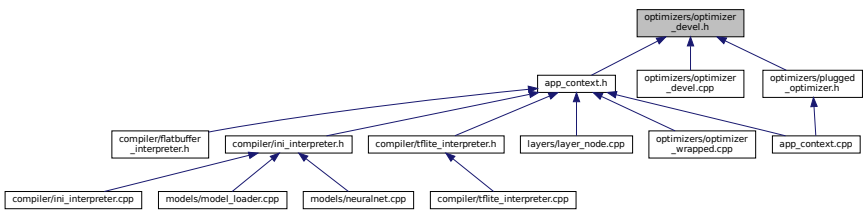
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.165 optimizers/optimizer_devel.h File Reference

This is Optimizer internal interface class.

This graph shows which files directly or indirectly include this file:



9.165.1 Detailed Description

This is Optimizer internal interface class.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

08 April 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Parichay Kapoor pk.kapoor@samsung.com

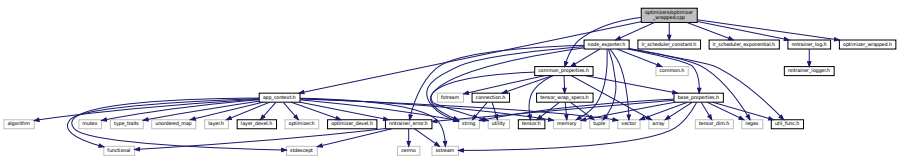
Bug No known bugs except for NYI items

9.166 optimizers/optimizer_wrapped.cpp File Reference

This is Optimizer Wrapped interface class.

```
#include <app_context.h>
#include <common_properties.h>
#include <lr_scheduler_constant.h>
#include <lr_scheduler_exponential.h>
#include <nntrainer_log.h>
#include <node_exporter.h>
#include <optimizer_wrapped.h>
```

Include dependency graph for optimizer_wrapped.cpp:



Namespaces

- nntrainer

Functions

- std::unique_ptr< OptimizerWrapped > nntrainer::createOptimizerWrapped (const ml::train::OptimizerType &type, const std::vector< std::string > &properties)

  Optimizer wrapped creator with constructor for optimizer.
- std::unique_ptr< OptimizerWrapped > nntrainer::createOptimizerWrapped (const std::string &type, const std::vector< std::string > &properties)

  Optimizer wrapped creator with constructor for optimizer.
- std::unique_ptr< OptimizerWrapped > nntrainer::createOptimizerWrapped (std::unique_ptr< OptimizerCore > &&opt, const std::vector< std::string > &properties)

  Optimizer wrapped creator with constructor for optimizer.

9.166.1 Detailed Description

This is Optimizer Wrapped interface class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

10 December 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

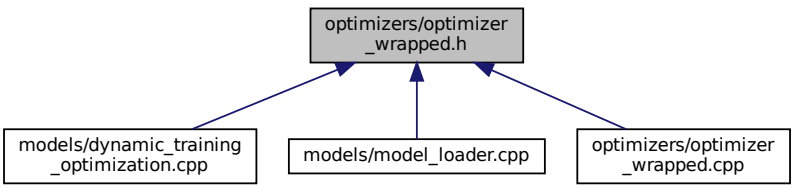
Bug No known bugs except for NYI items

wraps the optimizer and learning rate scheduler together

9.167 optimizers/optimizer_wrapped.h File Reference

This is Optimizer Wrapped interface class.

This graph shows which files directly or indirectly include this file:



9.167.1 Detailed Description

This is Optimizer Wrapped interface class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

10 December 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

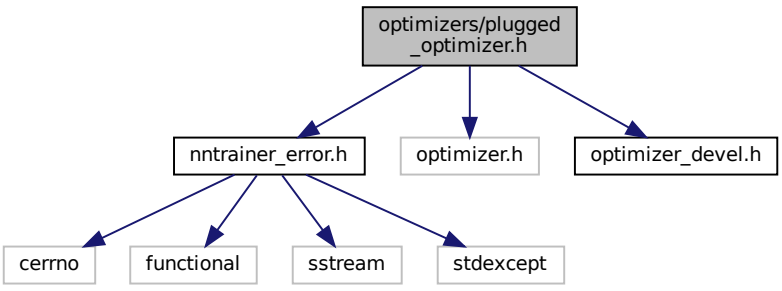
wraps the optimizer and learning rate scheduler together

9.168 optimizers/plugged_optimizer.h File Reference

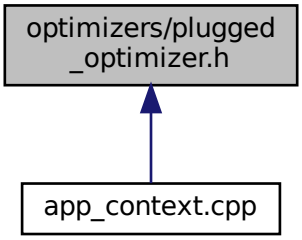
This file contains a wrapper for a plugged optimizer, INTERNAL USE ONLY.

```
#include <nntrainer_error.h>
#include <optimizer.h>
#include <optimizer_devel.h>
```

Include dependency graph for plugged_optimizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::internal::PluggedOptimizer

  Plugged optimizer class.

Namespaces

- nntrainer

9.168.1 Detailed Description

This file contains a wrapper for a plugged optimizer, INTERNAL USE ONLY.

Copyright (C) 2021 Parichay Kpaoor pk.kapoor@samsung.com

Date

1 June 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Jihoon Lee jhoon.it.lee@samsung.com

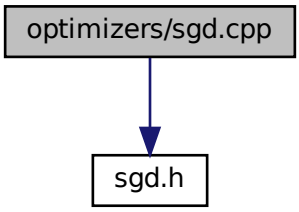
Bug No known bugs except for NYI items

9.169 optimizers/sgd.cpp File Reference

This is the SGD optimizer.

```
#include <sgd.h>
```

Include dependency graph for sgd.cpp:



Namespaces

- nntrainer

9.169.1 Detailed Description

This is the SGD optimizer.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

6 October 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

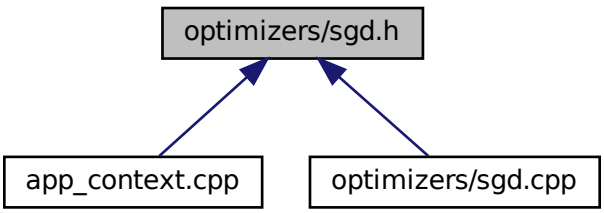
Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

9.170 optimizers/sgd.h File Reference

This is the SGD optimizer.

This graph shows which files directly or indirectly include this file:



9.170.1 Detailed Description

This is the SGD optimizer.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

6 October 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Parichay Kapoor pk.kapoor@samsung.com

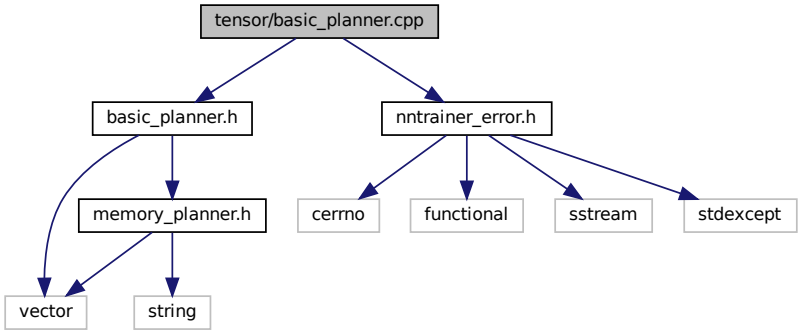
Bug No known bugs except for NYI items

9.171 tensor/basic_planner.cpp File Reference

This is Naive Memory Planner.

```
#include <basic_planner.h>
#include <nntrainer_error.h>
```

Include dependency graph for basic_planner.cpp:



Namespaces

- nntrainer

9.171.1 Detailed Description

This is Naive Memory Planner.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

11 August 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

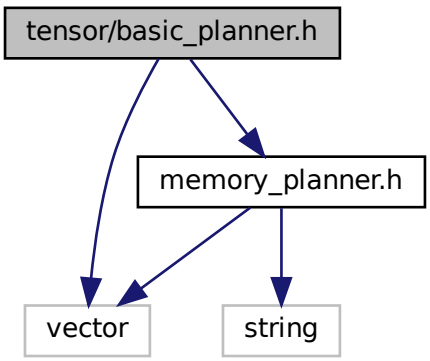
Bug No known bugs except for NYI items

9.172 tensor/basic_planner.h File Reference

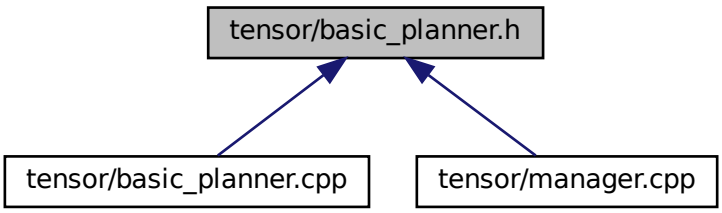
This is Naive Memory Planner.

```
#include <vector>
#include <memory_planner.h>
```

Include dependency graph for basic_planner.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::BasicPlanner

  Basic Memory Planner provides the basic plan for memory layout.

Namespaces

- nntrainer

9.172.1 Detailed Description

This is Naive Memory Planner.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

11 August 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

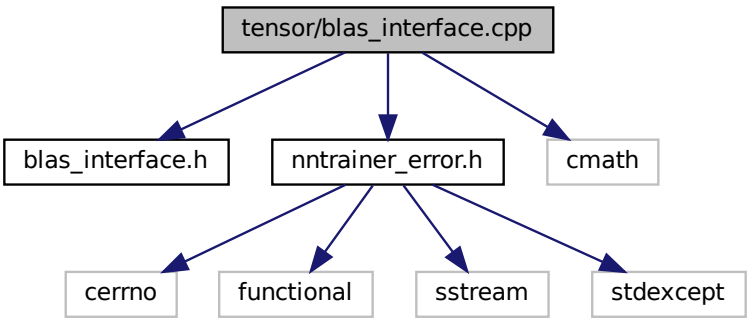
Bug No known bugs except for NYI items

9.173 tensor/blas_interface.cpp File Reference

This is dummy header for blas support.

```
#include <blas_interface.h>
#include <nntrainer_error.h>
#include <cmath>
```

Include dependency graph for blas_interface.cpp:



Namespaces

- nntrainer

Macros

- #define **sgemv_loop**(ci, cj, cM, cN)

Functions

- static void **nntrainer::saxpy_raw** (const unsigned int N, const float alpha, const float ∗X, const int incX, float ∗Y, const int incY)
- static void **nntrainer::sgemv_raw** (CBLAS_ORDER order, CBLAS_TRANSPOSE TransA, const unsigned int M, const unsigned int N, const float alpha, const float ∗A, const unsigned int lda, const float ∗X, const int incX, const float beta, float ∗Y, const int incY)
- static float **nntrainer::sdot_raw** (const unsigned int N, const float ∗X, const unsigned int incX, const float ∗Y, const unsigned int incY)
- static void **nntrainer::scopy_raw** (const unsigned int N, const float ∗X, const int incX, float ∗Y, const int incY)
- static void **nntrainer::sscal_raw** (const unsigned int N, const float alpha, float ∗X, const int incX)
- static float **nntrainer::snrm2_raw** (const unsigned int N, const float ∗X, const int incX)
- static void **nntrainer::sgemm_raw** (CBLAS_ORDER order, CBLAS_TRANSPOSE TransA, CBLAS_TRA↩ NSPOSE TransB, const unsigned int M, const unsigned int N, const unsigned int K, const float alpha, const float ∗A, const unsigned int lda, const float ∗B, const unsigned int ldb, const float beta, float ∗C, const unsigned int ldc)
- static unsigned int **nntrainer::isamax_raw** (const unsigned int N, const float ∗X, const int incX)
- void **nntrainer::saxpy** (const unsigned int N, const float alpha, const float ∗X, const int incX, float ∗Y, const int incY)
- void **nntrainer::sgemm** (CBLAS_ORDER order, CBLAS_TRANSPOSE TransA, CBLAS_TRANSPOSE TransB, const unsigned int M, const unsigned int N, const unsigned int K, const float alpha, const float ∗A, const unsigned int lda, const float ∗B, const unsigned int ldb, const float beta, float ∗C, const unsigned int ldc)
- void **nntrainer::scopy** (const unsigned int N, const float ∗X, const int incX, float ∗Y, const int incY)
- void **nntrainer::sscal** (const int N, const float alpha, float ∗X, const int incX)
- float **nntrainer::snrm2** (const int N, const float ∗X, const int incX)
- float **nntrainer::sdot** (const unsigned int N, const float ∗X, const unsigned int incX, const float ∗Y, const unsigned int incY)
- void **nntrainer::sgemv** (CBLAS_ORDER order, CBLAS_TRANSPOSE TransA, const unsigned int M, const unsigned int N, const float alpha, const float ∗A, const unsigned int lda, const float ∗X, const int incX, const float beta, float ∗Y, const int incY)
- unsigned int **nntrainer::isamax** (const unsigned int N, const float ∗X, const int incX)

9.173.1 Detailed Description

This is dummy header for blas support.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

28 Aug 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.173.2 Macro Definition Documentation

9.173.2.1 sgemv_loop

```
#define sgemv_loop(
              ci,
              cj,
              cM,
              cN )
```

Value:

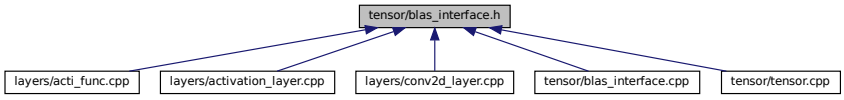
```
  do {                                          \
    double y0;                                  \
    unsigned int i, j;                          \
    for (ci = 0; ci != cM; ci++) {              \
      y0 = Y[ci * incy] * beta;                 \
      for (cj = 0; cj != cN; cj++)              \
        y0 += A[i + j * lda] * X[cj * incx];    \
      Y[ci * incy] = y0;                        \
    }                                           \
  } while (0);
```

Definition at line 19 of file blas_interface.cpp.

9.174 tensor/blas_interface.h File Reference

This is dummy header for blas support.

This graph shows which files directly or indirectly include this file:



9.174.1 Detailed Description

This is dummy header for blas support.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

28 Aug 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

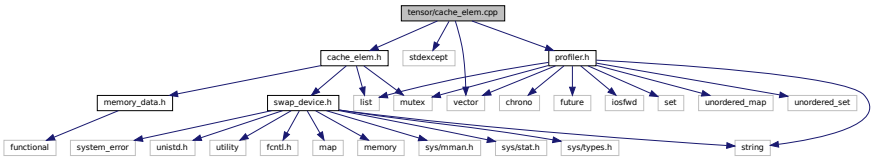
Bug No known bugs except for NYI items

9.175 tensor/cache_elem.cpp File Reference

Cache elem class.

```
#include "cache_elem.h"
#include <stdexcept>
#include <vector>
#include <profiler.h>
```

Include dependency graph for cache_elem.cpp:



Namespaces

- nntrainer

9.175.1 Detailed Description

Cache elem class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

28 Nov 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

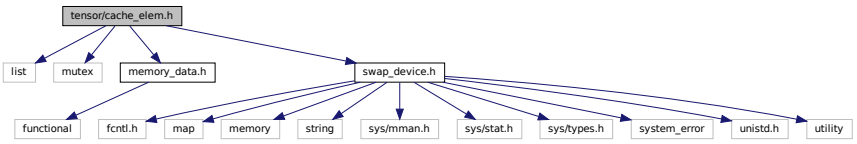
Bug No known bugs except for NYI items

9.176 tensor/cache_elem.h File Reference

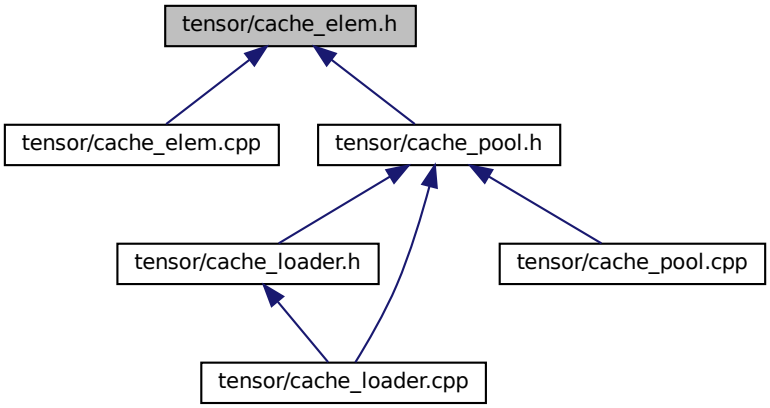
Cache elem class.

```
#include <list>
#include <mutex>
#include <memory_data.h>
#include <swap_device.h>
```

Include dependency graph for cache_elem.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::CacheElem

  Cache element containing swap address.

Namespaces

- nntrainer

Enumerations

- enum nntrainer::CachePolicy {
  nntrainer::WRITE_BACK = 0b0001, nntrainer::NO_WRITE_BACK = 0b0010, nntrainer::READ_CONSIST = 0b0100, nntrainer::NO_READ_CONSIST = 0b1000,
  nntrainer::ALWAYS_SYNCED, nntrainer::TEMPORAL, nntrainer::FIRST_LAST_SKIP = 0b10000, nntrainer::ITERATION_CON = (FIRST_LAST_SKIP | ALWAYS_SYNCED) }

9.176.1 Detailed Description

Cache elem class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

28 Nov 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

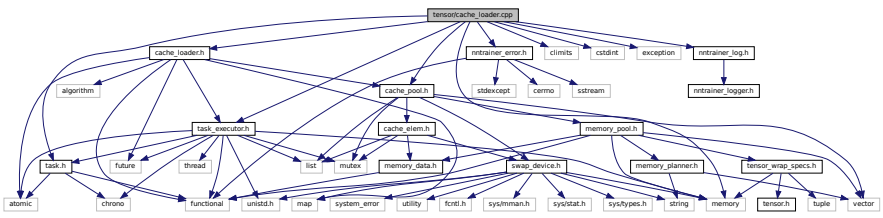
Bug No known bugs except for NYI items

9.177 tensor/cache_loader.cpp File Reference

Cache loader class.

```
#include "cache_loader.h"
#include "task.h"
#include "task_executor.h"
#include <cache_pool.h>
#include <climits>
#include <cstdint>
#include <exception>
#include <memory>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
```

Include dependency graph for cache_loader.cpp:



Namespaces

- nntrainer

9.177.1 Detailed Description

Cache loader class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

10 Nov 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

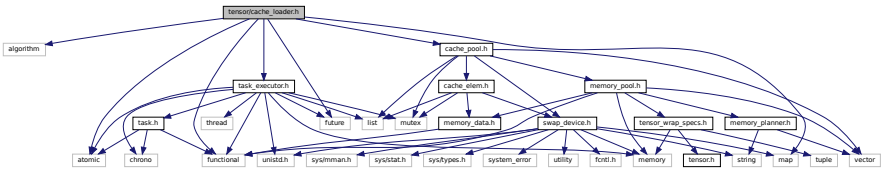
Bug No known bugs except for NYI items

9.178 tensor/cache_loader.h File Reference

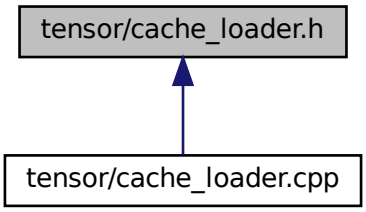
Cache loader class.

```
#include <algorithm>
#include <atomic>
#include <functional>
#include <future>
#include <map>
#include <cache_pool.h>
#include <task_executor.h>
```

Include dependency graph for cache_loader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::CacheLoader

  Cache loader from swap device.

Namespaces

- nntrainer

9.178.1 Detailed Description

Cache loader class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

10 Nov 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

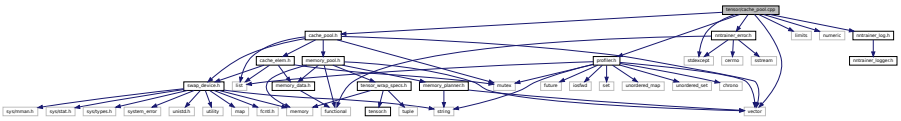
Bug No known bugs except for NYI items

9.179 tensor/cache_pool.cpp File Reference

Cache pool class inherited from memory pool.

```
#include "cache_pool.h"
#include <limits>
#include <numeric>
#include <stdexcept>
#include <vector>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <profiler.h>
```

Include dependency graph for cache_pool.cpp:



Namespaces

- nntrainer

9.179.1 Detailed Description

Cache pool class inherited from memory pool.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

01 July 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

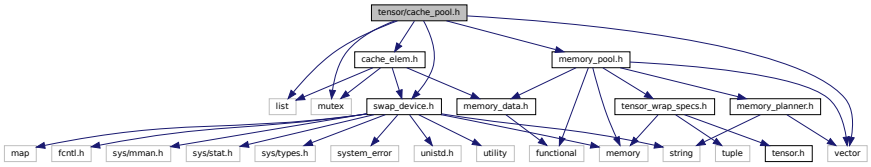
Bug No known bugs except for NYI items

9.180 tensor/cache_pool.h File Reference

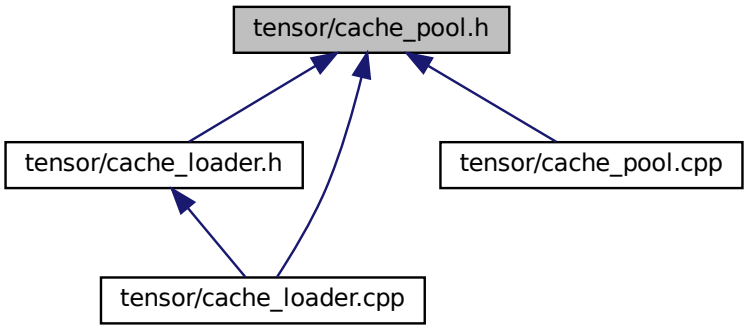
Cache pool class inherited from memory pool.

```
#include <list>
#include <mutex>
#include <vector>
#include <cache_elem.h>
#include <memory_pool.h>
#include <swap_device.h>
```

Include dependency graph for cache_pool.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::CachePool

  Cache memory with fixed size utilizing swap device.

Namespaces

- nntrainer

9.180.1 Detailed Description

Cache pool class inherited from memory pool.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

01 July 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

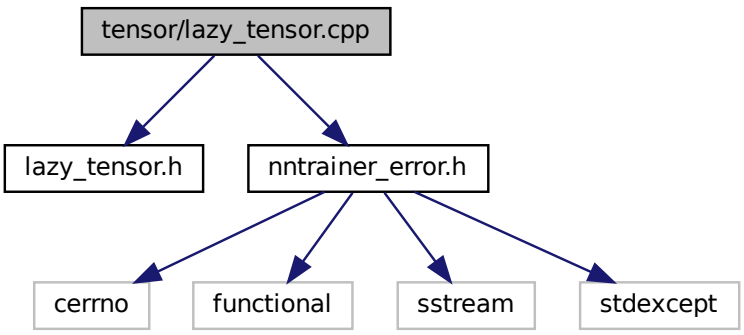
Bug No known bugs except for NYI items

9.181 tensor/lazy_tensor.cpp File Reference

A lazy evaluation calculator for tensors.

```
#include <lazy_tensor.h>
#include <nntrainer_error.h>
```

Include dependency graph for lazy_tensor.cpp:



Namespaces

- nntrainer

9.181.1 Detailed Description

A lazy evaluation calculator for tensors.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

05 Jun 2020

See also

https://github.com/nnstreamer/nntrainer

Author

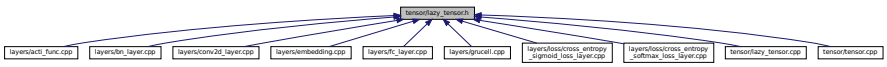
Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

9.182 tensor/lazy_tensor.h File Reference

A lazy evaluation calculator for tensors.

This graph shows which files directly or indirectly include this file:



9.182.1 Detailed Description

A lazy evaluation calculator for tensors.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

05 Jun 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

9.183 tensor/manager.cpp File Reference

This is NNtrainer manager for all weights, i/o and intermediate tensors.

```
#include <fcntl.h>
#include <functional>
#include <limits>
#include <stdexcept>
#include <sys/mman.h>
#include <sys/stat.h>
#include <unistd.h>
#include <vector>
#include <activation_layer.h>
#include <basic_planner.h>
#include <bn_layer.h>
#include <graph_node.h>
#include <grucell.h>
#include <layer_node.h>
#include <layer_normalization_layer.h>
#include <loss/cross_entropy_sigmoid_loss_layer.h>
#include <loss/cross_entropy_softmax_loss_layer.h>
#include <loss/mse_loss_layer.h>
#include <manager.h>
#include <multiout_layer.h>
#include <nntrainer_log.h>
#include <optimized_v1_planner.h>
#include <optimized_v2_planner.h>
#include <optimized_v3_planner.h>
#include <tensor_pool.h>
#include <tensor_wrap_specs.h>
#include <util_func.h>
#include <var_grad.h>
```

Include dependency graph for manager.cpp:

Namespaces

- nntrainer

Functions

- static Tensor * **nntrainer::requestTensor_** (const TensorSpecV2 &spec, const GraphNode::ExecutionOrder &exec_order, const std::string &scope, TensorPool &tp, bool expose, bool trainable)

9.183.1 Detailed Description

This is NNtrainer manager for all weights, i/o and intermediate tensors.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

2 Dec 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Jihoon Lee jhoon.it.lee@samsung.com

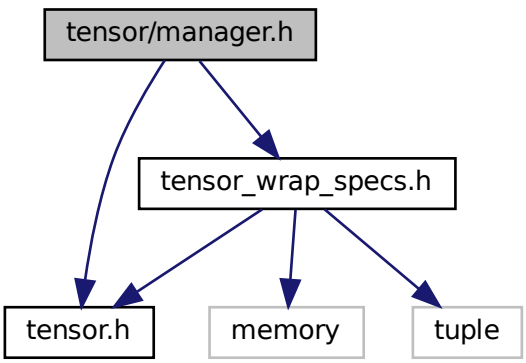
Bug No known bugs except for NYI items

9.184 tensor/manager.h File Reference

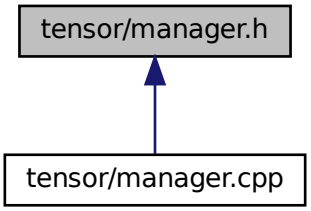
This is NNtrainer manager for all weights, i/o and intermediate tensors.

```
#include "tensor.h"
#include "tensor_wrap_specs.h"
```

Include dependency graph for manager.h:



This graph shows which files directly or indirectly include this file:



9.184.1 Detailed Description

This is NNtrainer manager for all weights, i/o and intermediate tensors.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

30 Nov 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

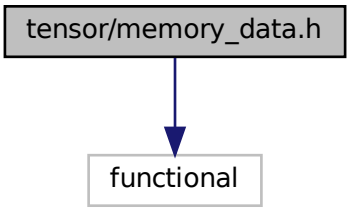
Manager assumes that the layer inouts are being tracked by the manager in the order of the execution. If the order is not maintained, then the optimizations cannot be performed and will result in wrong values.

9.185 tensor/memory_data.h File Reference

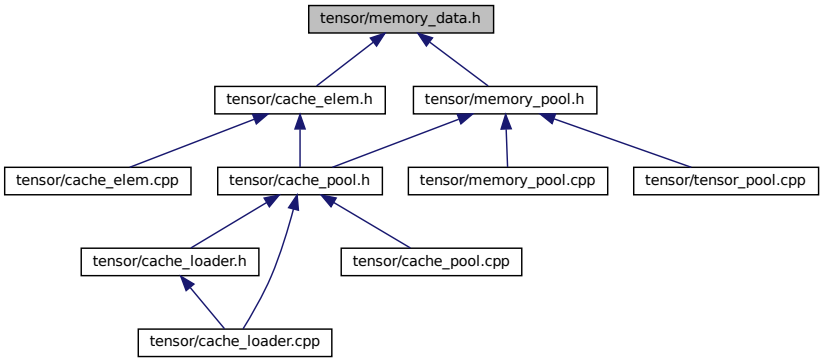
MemoryData class.

```
#include <functional>
```

Include dependency graph for memory_data.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::MemoryData< T >

  MemoryData Class.

Namespaces

- nntrainer

Typedefs

- using **nntrainer::MemoryDataValidateCallback** = std::function< void(unsigned int)>

9.185.1 Detailed Description

MemoryData class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

14 Oct 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

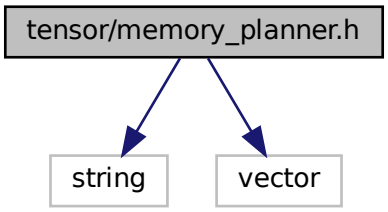
Bug No known bugs except for NYI items

9.186 tensor/memory_planner.h File Reference

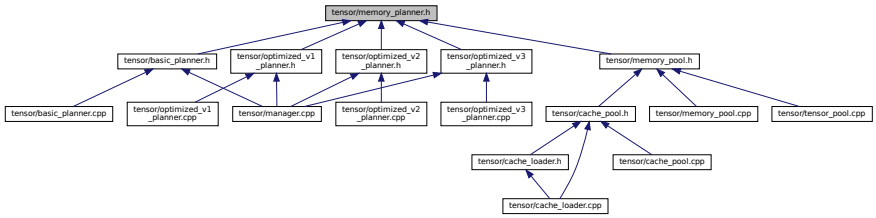
This is interface for the Memory Planner.

```
#include <string>
#include <vector>
```

Include dependency graph for memory_planner.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::MemoryPlanner

  Memory Planner provides the plan/strategy to allocate the memory.

Namespaces

- nntrainer

9.186.1 Detailed Description

This is interface for the Memory Planner.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

10 August 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

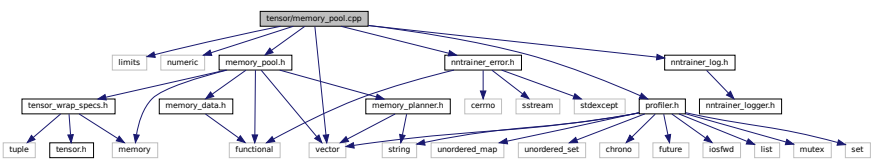
Bug No known bugs except for NYI items

9.187 tensor/memory_pool.cpp File Reference

This is Memory Pool Class.

```
#include <limits>
#include <numeric>
#include <vector>
#include <memory_pool.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <profiler.h>
```

Include dependency graph for memory_pool.cpp:



Namespaces

- nntrainer

Functions

- template<typename T >
  static bool nntrainer::overlap (T s1, T e1, T s2, T e2)
  check if the two given intervals overlap

9.187.1 Detailed Description

This is Memory Pool Class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

11 August 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

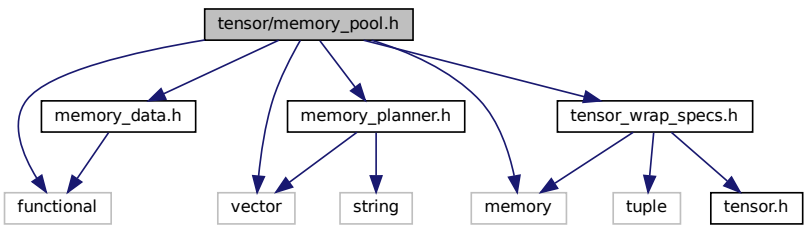
Bug No known bugs except for NYI items

9.188 tensor/memory_pool.h File Reference

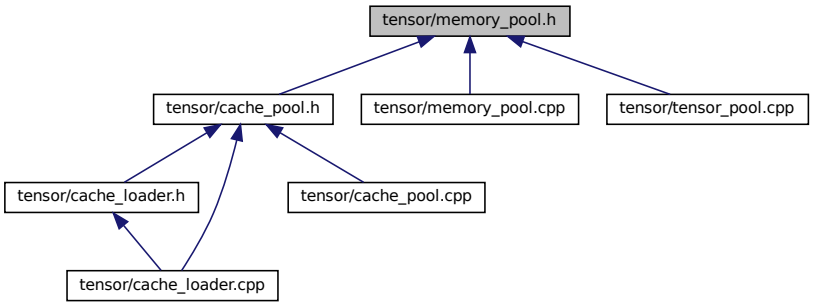
This is Memory Pool Class.

```
#include <functional>
#include <memory>
#include <vector>
#include <memory_data.h>
#include <memory_planner.h>
#include <tensor_wrap_specs.h>
```

Include dependency graph for memory_pool.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::MemoryPool

  Memory Pool provides a common pool for all the tensor memory.

Namespaces

- nntrainer

9.188.1 Detailed Description

This is Memory Pool Class.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

10 August 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Bug No known bugs except for NYI items

Todo Support an external allocator for different backends and alignment

Support releaseMemory(token) - this need not release actual memory until deallocate

Support maximum memory size for the memory pool as an argument

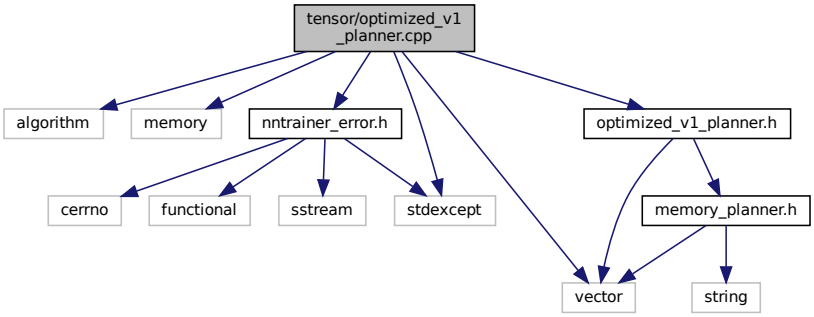
support late memory request without optimization

9.189 tensor/optimized_v1_planner.cpp File Reference

This is Optimized V1 Memory Planner.

```
#include <algorithm>
#include <memory>
#include <nntrainer_error.h>
#include <stdexcept>
#include <vector>
#include <optimized_v1_planner.h>
```

Include dependency graph for optimized_v1_planner.cpp:



Classes

- struct nntrainer::MemoryRequest

  Memory Request data structure clubbing all the requests.

Namespaces

- nntrainer

Functions

- static void nntrainer::validateIntervalOverlap (const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, const std::vector< size_t > &memory_size, const std::vector< size_t > &memory_offset, size_t memory_req)

  check if validate interval is overlapping in a very naive way.

9.189.1 Detailed Description

This is Optimized V1 Memory Planner.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

3 September 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

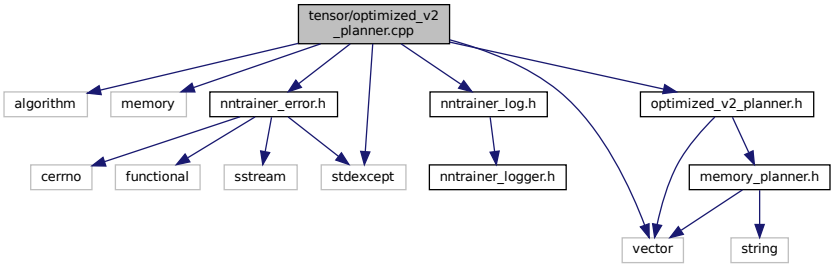
Bug No known bugs except for NYI items

9.190 tensor/optimized_v2_planner.cpp File Reference

This is Optimized V2 Memory Planner.

```
#include <algorithm>
#include <memory>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <stdexcept>
#include <vector>
#include <optimized_v2_planner.h>
```

Include dependency graph for optimized_v2_planner.cpp:



Classes

- struct nntrainer::MemoryRequest

  Memory Request data structure clubbing all the requests.
- struct nntrainer::WGradMemoryRequest

  Memory Request data structure clubbing for the weight gradient requests.

Namespaces

- nntrainer

Functions

- static void nntrainer::validateIntervalOverlap (const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, const std::vector< size_t > &memory_size, const std::vector< size_t > &memory_offset, size_t memory_req)

  check if validate interval is overlapping in a very naive way.

9.190.1 Detailed Description

This is Optimized V2 Memory Planner.

Copyright (C) 2022 Jijoong Moon jijoong.moon@samsung.com

Date

29 December 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Jijoong Moon jijoong.moon@samsung.com

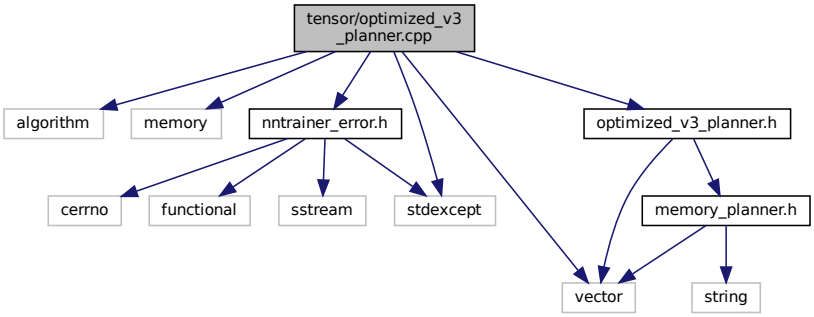
Bug No known bugs except for NYI items

9.191 tensor/optimized_v3_planner.cpp File Reference

This is Optimized V3 Memory Planner.

```
#include <algorithm>
#include <memory>
#include <nntrainer_error.h>
#include <stdexcept>
#include <vector>
#include <optimized_v3_planner.h>
```

Include dependency graph for optimized_v3_planner.cpp:



Classes

- struct nntrainer::MemoryRequest

  Memory Request data structure clubbing all the requests.

Namespaces

- nntrainer

Functions

- static size_t nntrainer::computeSpace (unsigned int exec_order, std::vector< MemoryRequest * > &sorted_req, std::vector< std::pair< size_t, size_t >> &vacant)
- static void nntrainer::validateIntervalOverlap (const std::vector< std::pair< unsigned int, unsigned int >> &memory_validity, const std::vector< size_t > &memory_size, const std::vector< size_t > &memory_offset, size_t memory_req)

  check if validate interval is overlapping in a very naive way.

9.191.1 Detailed Description

This is Optimized V3 Memory Planner.

Copyright (C) 2023 Jijoong Moon jijoong.moon@samsung.com

Date

2 January 2023

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

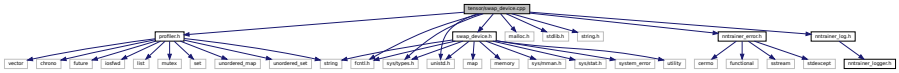
Bug No known bugs except for NYI items

9.192 tensor/swap_device.cpp File Reference

Swap device class implementation.

```
#include <fcntl.h>
#include <malloc.h>
#include <profiler.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <swap_device.h>
```

Include dependency graph for swap_device.cpp:



Namespaces

- nntrainer

9.192.1 Detailed Description

Swap device class implementation.

Swap device class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

01 July 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

Bug No known bugs except for NYI items

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

01 July 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

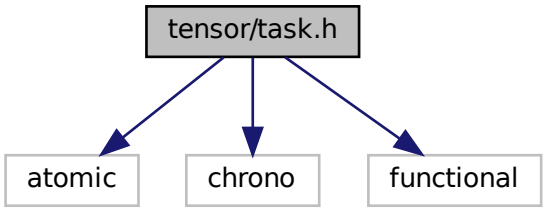
Bug No known bugs except for NYI items

9.193 tensor/task.h File Reference

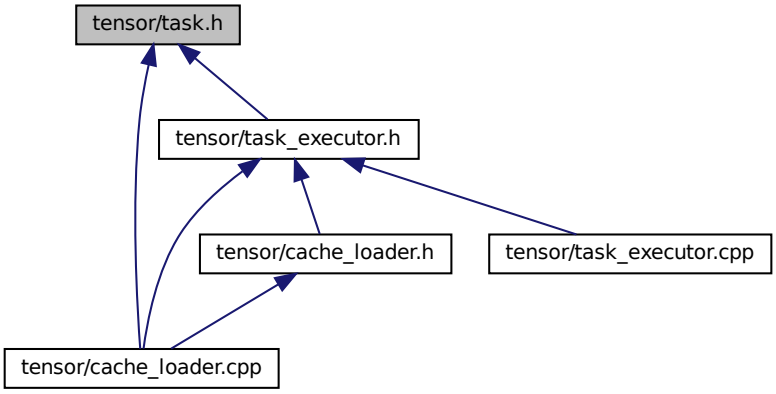
Task class.

```
#include <atomic>
#include <chrono>
#include <functional>
```

Include dependency graph for task.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::Task

  task class
- class nntrainer::TaskAsync< T >

  Async task class.

Namespaces

- nntrainer

9.193.1 Detailed Description

Task class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

04 Nov 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

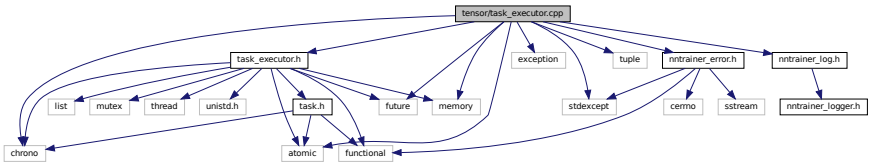
Bug No known bugs except for NYI items

9.194 tensor/task_executor.cpp File Reference

Task executor class.

```
#include "task_executor.h"
#include <atomic>
#include <chrono>
#include <exception>
#include <future>
#include <memory>
#include <stdexcept>
#include <tuple>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
```

Include dependency graph for task_executor.cpp:



Namespaces

- nntrainer

9.194.1 Detailed Description

Task executor class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

04 Nov 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

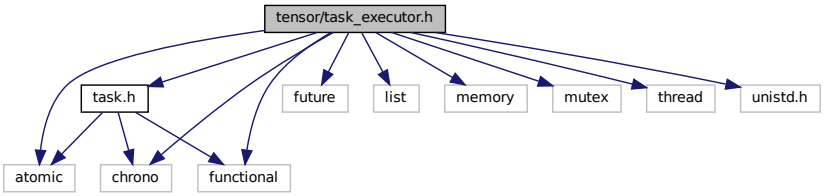
Bug No known bugs except for NYI items

9.195 tensor/task_executor.h File Reference

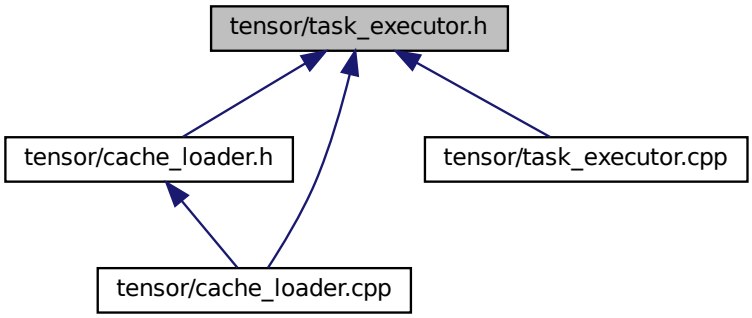
Task executor class.

```
#include <atomic>
#include <chrono>
#include <functional>
#include <future>
#include <list>
#include <memory>
#include <mutex>
#include <thread>
#include <unistd.h>
#include <task.h>
```

Include dependency graph for task_executor.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::TaskExecutor

  task executor class

Namespaces

- nntrainer

9.195.1 Detailed Description

Task executor class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

04 Nov 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

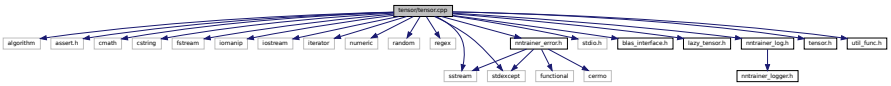
Bug No known bugs except for NYI items

9.196 tensor/tensor.cpp File Reference

This is Tensor class for calculation.

```
#include <algorithm>
#include <assert.h>
#include <cmath>
#include <cstring>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <numeric>
#include <random>
#include <regex>
#include <sstream>
#include <stdexcept>
#include <stdio.h>
#include <blas_interface.h>
#include <lazy_tensor.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <tensor.h>
#include <util_func.h>
```

Include dependency graph for tensor.cpp:



Classes

- struct nntrainer::Tensor::BroadcastInfo
- class nntrainer::SrcSharedTensor

  Source of the shared tensor.

Namespaces

- nntrainer

Macros

- #define **transposeloop**(cl, ci, cj, ck, sl, si, sj, sk)
- #define **CREATE_IF_EMPTY_DIMS**(tensor, ...)

Functions

- std::ostream & **nntrainer::operator**<< (std::ostream &out, Tensor const &m)

Variables

- static auto **nntrainer::rng**

9.196.1 Detailed Description

This is Tensor class for calculation.

Date

04 December 2019

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.196.2 Macro Definition Documentation

9.196.2.1 CREATE_IF_EMPTY_DIMS

```
#define CREATE_IF_EMPTY_DIMS(
              tensor,
              ... )
```

Value:

```
  do {                                    \
    if (tensor.empty())                   \
      tensor = Tensor(__VA_ARGS__);       \
  } while (0);
```

Definition at line 60 of file tensor.cpp.

9.196.2.2 transposeloop

```
#define transposeloop(
                cl,
                ci,
                cj,
                ck,
                sl,
                si,
                sj,
                sk )
```

Value:

```
  do {                                                                  \
    unsigned int i, j, k, l;                                            \
    int inidx = 0, outidx = 0;                                          \
    for (cl = 0; cl < sl; cl++)                                         \
      for (ci = 0; ci < si; ci++)                                       \
        for (cj = 0; cj < sj; cj++)                                     \
          for (ck = 0; ck < sk; ck++) {                                 \
            outidx = si * sj * sk * cl + sj * sk * ci + sk * cj + ck;   \
            inidx = l * SI * SJ * SK + i * SJ * SK + j * SK + k;        \
            outptr[outidx] = inptr[inidx];                              \
          }                                                             \
  } while (0);
```

Definition at line 46 of file tensor.cpp.

9.197 tensor/tensor.h File Reference

This is Tensor class for calculation.

This graph shows which files directly or indirectly include this file:

9.197.1 Detailed Description

This is Tensor class for calculation.

Date

04 December 2019

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

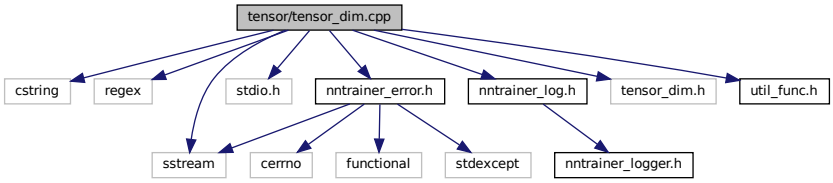
Todo deprecate new tensor allocation for out of place operations.

9.198 tensor/tensor_dim.cpp File Reference

This is Tensor Dimension Class.

```
#include <cstring>
#include <regex>
#include <sstream>
#include <stdio.h>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <tensor_dim.h>
#include <util_func.h>
```

Include dependency graph for tensor_dim.cpp:



Functions

- void **ml::train::swap** (TensorDim &lhs, TensorDim &rhs) noexcept
- std::ostream & **ml::train::operator**<< (std::ostream &out, TensorDim const &d)

9.198.1 Detailed Description

This is Tensor Dimension Class.

Copyright (C) 2020 Jijoong Moon jijoong.moon@samsung.com

Date

22 May 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

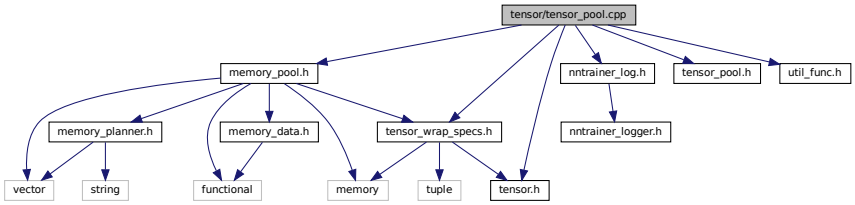
Bug No known bugs except for NYI items

9.199 tensor/tensor_pool.cpp File Reference

This is TensorPool for all requested tensors.

```
#include <memory_pool.h>
#include <nntrainer_log.h>
#include <tensor.h>
#include <tensor_pool.h>
#include <tensor_wrap_specs.h>
#include <util_func.h>
```

Include dependency graph for tensor_pool.cpp:



Namespaces

- nntrainer

9.199.1 Detailed Description

This is TensorPool for all requested tensors.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

19 Aug 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

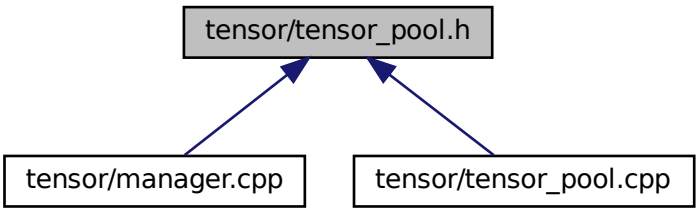
Todo add checks for request/updates that finalize is not done

check before allocate that finalize is done

9.200 tensor/tensor_pool.h File Reference

This is TensorPool for all requested tensors.

This graph shows which files directly or indirectly include this file:



9.200.1 Detailed Description

This is TensorPool for all requested tensors.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

18 Aug 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

Jihoon Lee jhoon.it.lee@samsung.com

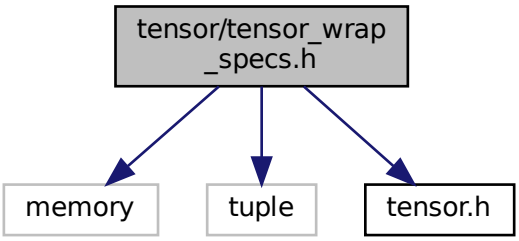
Bug No known bugs except for NYI items

9.201 tensor/tensor_wrap_specs.h File Reference

This is specs for various tensor wrappers.

```
#include <memory>
#include <tuple>
#include <tensor.h>
```

Include dependency graph for tensor_wrap_specs.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct nntrainer::TensorSpecV2

  Tensor Specification which describes how this tensor should be allocated and managed.
- struct nntrainer::VarGradSpecV2

  variable + gradient specification
- struct nntrainer::WeightSpecV2

  weight specification

Namespaces

- nntrainer

Typedefs

- typedef std::tuple< TensorDim, Tensor::Initializer, WeightRegularizer, float, float, float, bool, const std::string > nntrainer::WeightSpec

  Specification of the Weight as a tensor wrapper.
- typedef std::tuple< TensorDim, Tensor::Initializer, bool, const std::string, TensorLifespan > nntrainer::VarGradSpec

  Specification of the Var_Grad (trainable tensor) as a tensor wrapper.

Enumerations

- enum nntrainer::WeightRegularizer { nntrainer::WeightRegularizer::L2NORM, nntrainer::WeightRegularizer::NONE, nntrainer::WeightRegularizer::UNKNOWN }

  Enumeration of Weight Regularizer.

- enum nntrainer::TensorLifespan { nntrainer::TensorLifespan::UNMANAGED = 0b000, nntrainer::TensorLifespan::FORWARD_FUNC_LIFESPAN = 0b001, nntrainer::TensorLifespan::CALC_DERIV_LIFESPAN = 0b010, nntrainer::TensorLifespan::CALC_GRAD_LIFESPAN = 0b100, nntrainer::TensorLifespan::CALC_AGRAD_LIFESPAN = 0b1000, nntrainer::TensorLifespan::CALC_GRAD_DERIV_LIFESPAN = 0b110, nntrainer::TensorLifespan::CALC_GRAD_DERIV_AGRAD_LIFESPAN = 0b1110, nntrainer::TensorLifespan::FORWA = 0b101, nntrainer::TensorLifespan::FORWARD_GRAD_AGRAD_LIFESPAN = 0b1101, nntrainer::TensorLifespan::FORWARD_DERIV_ = 0b011, nntrainer::TensorLifespan::BACKWARD_FUNC_LIFESPAN, nntrainer::TensorLifespan::ITERATION_LIFESPAN = 0b1111, nntrainer::TensorLifespan::EPOCH_LIFESPAN = 0b11111, nntrainer::TensorLifespan::MAX_LIFESPAN = 0b111111 }

  define the lifespan of the given tensor to reduce peak memory

9.201.1 Detailed Description

This is specs for various tensor wrappers.

Copyright (C) 2021 Parichay Kapoor pk.kapoor@samsung.com

Date

26 July 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

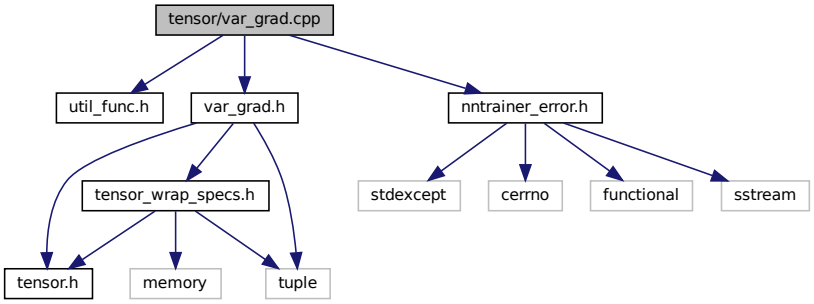
Bug No known bugs except for NYI items

9.202 tensor/var_grad.cpp File Reference

This is Var_Grad Class for Neural Network.

```
#include <util_func.h>
#include <var_grad.h>
```

`#include <nntrainer_error.h>`
Include dependency graph for var_grad.cpp:



Namespaces

- nntrainer

9.202.1 Detailed Description

This is Var_Grad Class for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

13 November 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

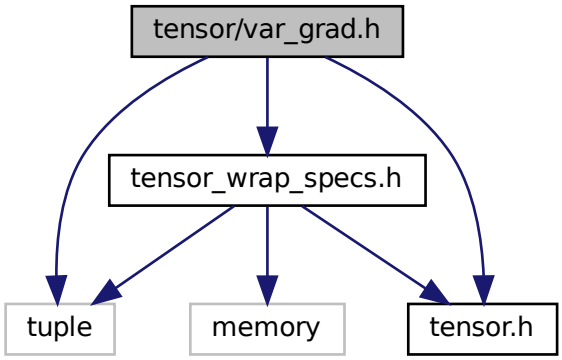
Bug No known bugs except for NYI items

9.203 tensor/var_grad.h File Reference

This is Var_Grad Class for Neural Network.

```
#include <tuple>
#include <tensor.h>
#include <tensor_wrap_specs.h>
```

Include dependency graph for var_grad.h:



This graph shows which files directly or indirectly include this file:

Classes

- class nntrainer::Var_Grad

  Variable with Gradient, and its corresponding need_gradient property.

Namespaces

- nntrainer

9.203.1 Detailed Description

This is Var_Grad Class for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

13 November 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

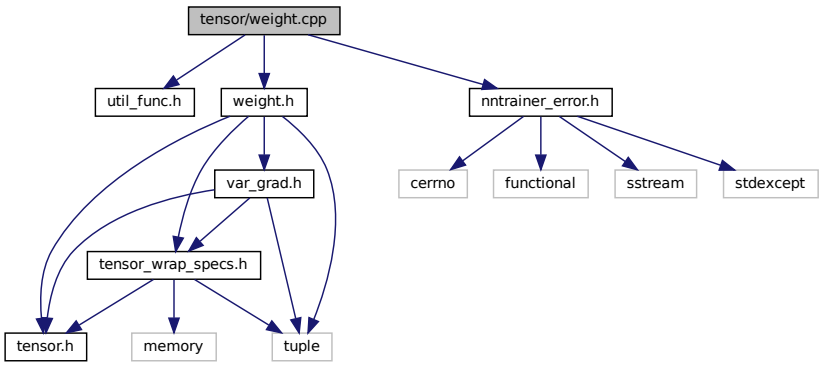
Bug No known bugs except for NYI items

9.204 tensor/weight.cpp File Reference

This is Weight Class for Neural Network.

```
#include <util_func.h>
#include <weight.h>
#include <nntrainer_error.h>
```

Include dependency graph for weight.cpp:



Namespaces

- nntrainer

9.204.1 Detailed Description

This is Weight Class for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

22 September 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

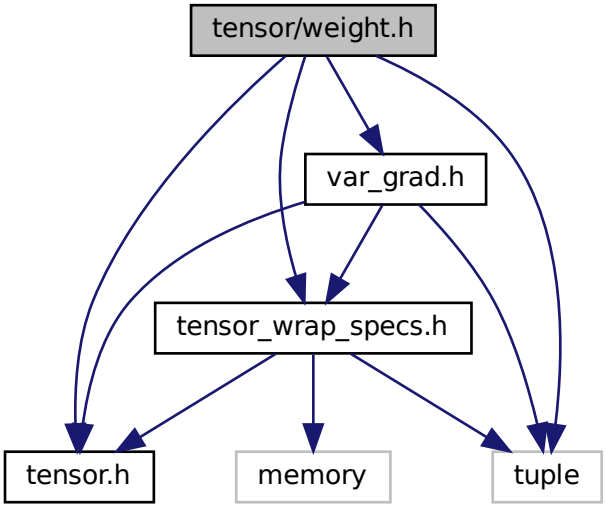
Bug No known bugs except for NYI items

9.205 tensor/weight.h File Reference

This is Weight Class for Neural Network.

```
#include <tuple>
#include <tensor.h>
#include <tensor_wrap_specs.h>
#include <var_grad.h>
```

Include dependency graph for weight.h:



This graph shows which files directly or indirectly include this file:

Classes

- class nntrainer::Weight

  Weight extends over Var_Grad with regularization & optimizer updates.

Namespaces

- nntrainer

9.205.1 Detailed Description

This is Weight Class for Neural Network.

Copyright (C) 2020 Parichay Kapoor pk.kapoor@samsung.com

Date

22 September 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Parichay Kapoor pk.kapoor@samsung.com

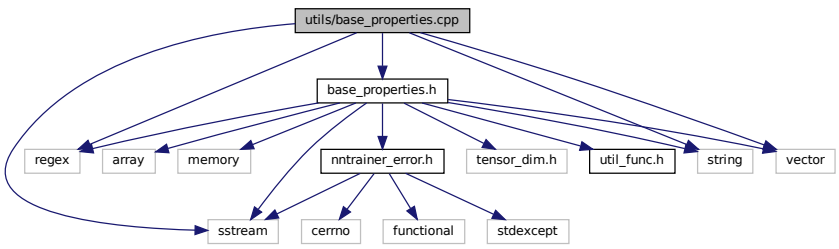
Bug No known bugs except for NYI items

9.206 utils/base_properties.cpp File Reference

Convenient property type definition for automated serialization.

```
#include <base_properties.h>
#include <regex>
#include <sstream>
#include <string>
#include <vector>
```

Include dependency graph for base_properties.cpp:



Namespaces

- nntrainer

9.206.1 Detailed Description

Convenient property type definition for automated serialization.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

03 May 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

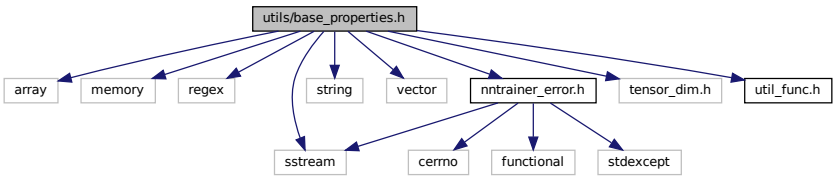
Bug No known bugs except for NYI items

9.207 utils/base_properties.h File Reference

Convenient property type definition for automated serialization.

```
#include <array>
#include <memory>
#include <regex>
#include <sstream>
#include <string>
#include <vector>
#include <nntrainer_error.h>
#include <tensor_dim.h>
#include <util_func.h>
```

Include dependency graph for base_properties.h:



This graph shows which files directly or indirectly include this file:

9.207.1 Detailed Description

Convenient property type definition for automated serialization.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

08 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

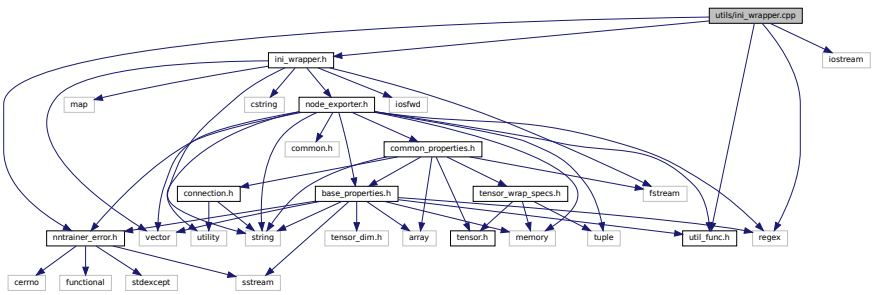
Bug No known bugs except for NYI items

9.208 utils/ini_wrapper.cpp File Reference

NNTrainer Ini Wrapper helps to save ini.

```
#include <ini_wrapper.h>
#include <regex>
#include <iostream>
#include <nntrainer_error.h>
#include <util_func.h>
```

Include dependency graph for ini_wrapper.cpp:



Namespaces

- nntrainer

9.208.1 Detailed Description

NNTrainer Ini Wrapper helps to save ini.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

08 April 2021

Note

this is to be used with ini_interpreter

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

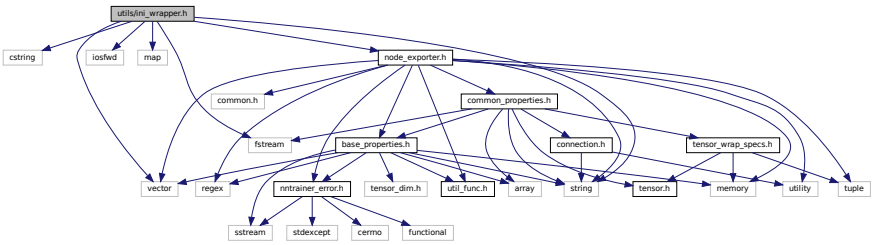
Bug No known bugs except for NYI items

9.209 utils/ini_wrapper.h File Reference

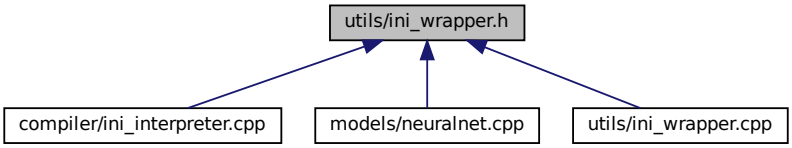
NNTrainer Ini Wrapper helps to save ini.

```
#include <cstring>
#include <fstream>
#include <iosfwd>
#include <map>
#include <string>
#include <vector>
#include <node_exporter.h>
```

Include dependency graph for ini_wrapper.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::IniSection

  IniSection class that maps to one ini section.
- class nntrainer::IniWrapper

  IniWrapper using IniSections.

Namespaces

- nntrainer

9.209.1 Detailed Description

NNTrainer Ini Wrapper helps to save ini.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

08 April 2021

Note

this is to be used with ini_interpreter

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

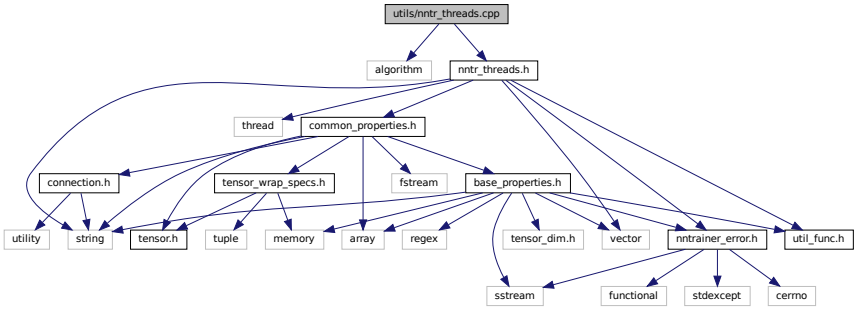
Bug No known bugs except for NYI items

9.210 utils/nntr_threads.cpp File Reference

Thread Management for NNTrainer.

```
#include <algorithm>
#include <nntr_threads.h>
```

Include dependency graph for nntr_threads.cpp:



Namespaces

- nntrainer

9.210.1 Detailed Description

Thread Management for NNTrainer.

Copyright (C) 2022 Jijoong Moon jijoong.moon@samsung.com

Date

07 July 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

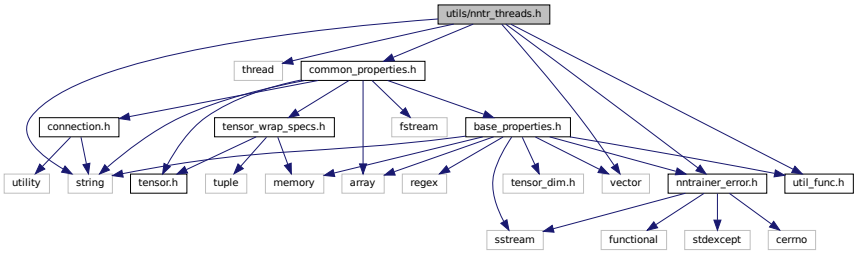
Bug No known bugs except for NYI items

9.211 utils/nntr_threads.h File Reference

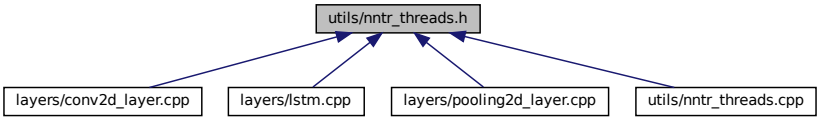
Thread Management for NNTrainer.

```
#include <string>
#include <thread>
#include <vector>
#include <common_properties.h>
#include <nntrainer_error.h>
#include <util_func.h>
```

Include dependency graph for nntr_threads.h:



This graph shows which files directly or indirectly include this file:



Classes

- class nntrainer::ParallelBatch

  ParallelBatch class to parallelize along batch direction.

Namespaces

- nntrainer

Typedefs

- typedef void(∗ **loop_cb**) (unsigned int start, unsigned int end, unsigned int pid, void ∗user_data)
- typedef std::function< std::remove_pointer< loop_cb >::type > **threaded_cb**

9.211.1 Detailed Description

Thread Management for NNTrainer.

Copyright (C) 2022 Jijoong Moon jijoong.moon@samsung.com

Date

07 July 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

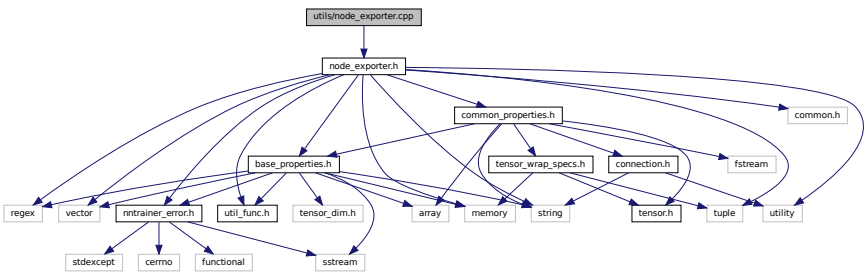
Bug No known bugs except for NYI items

9.212 utils/node_exporter.cpp File Reference

NNTrainer Node exporter.

```
#include <node_exporter.h>
```

Include dependency graph for node_exporter.cpp:



Namespaces

- nntrainer

Functions

- template<> std::unique_ptr< std::vector< std::pair< std::string, std::string > > > **nntrainer::Exporter::getResult< ml::train::ExportMethods::METHOD_STRINGVECTOR >** ()

Variables

- constexpr const unsigned int **nntrainer::CONV2D_DIM** = 2
- constexpr const unsigned int **nntrainer::POOLING2D_DIM** = 2

9.212.1 Detailed Description

NNTrainer Node exporter.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

Bug No known bugs except for NYI items

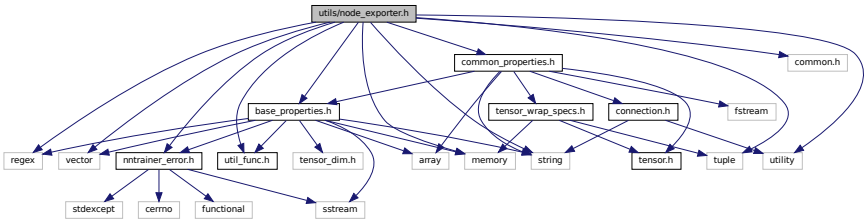
9.213 utils/node_exporter.h File Reference

NNTrainer Node exporter.

```
#include <memory>
#include <regex>
#include <string>
#include <tuple>
#include <utility>
#include <vector>
#include <base_properties.h>
#include <common.h>
#include <common_properties.h>
#include <nntrainer_error.h>
```

```
#include <util_func.h>
```

Include dependency graph for node_exporter.h:



This graph shows which files directly or indirectly include this file:

Classes

- class nntrainer::Exporter

  Exporter class helps to exports the node information in a predefined way. because each method will require complete different methods, this class exploits visitor pattern to make a custom defined saving method.

Namespaces

- nntrainer

Functions

- template<size_t I = 0, typename Callable , typename... Ts>
  std::enable_if< I==sizeof...(Ts), void >::type nntrainer::iterate_prop (Callable &&c, const std::tuple< Ts... > &tup)

  base case of iterate_prop, iterate_prop iterates the given tuple

- template<size_t I = 0, typename Callable , typename... Ts>
  std::enable_if<(I< sizeof...(Ts)), void >::type nntrainer::iterate_prop (Callable &&c, const std::tuple< Ts... > &tup)

  base case of iterate_prop, iterate_prop iterates the given tuple

- template<size_t I = 0, typename Callable , typename... Ts>
  std::enable_if<(I< sizeof...(Ts)), void >::type nntrainer::iterate_prop (Callable &&c, std::tuple< Ts... > &tup)

  <size_t I = 0, typename Callable, typename... Ts>

- template<typename Tuple >
  std::vector< std::string > nntrainer::loadProperties (const std::vector< std::string > &string_vector, Tuple &&props)

  load property from the api formatted string ({"key=value", "key1=value1"})

9.213.1 Detailed Description

NNTrainer Node exporter.

Copyright (C) 2021 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 April 2021

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

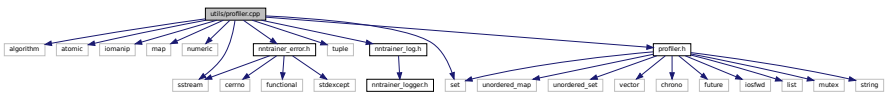
Bug No known bugs except for NYI items

9.214 utils/profiler.cpp File Reference

Profiler related codes to be used to benchmark things.

```
#include <algorithm>
#include <atomic>
#include <iomanip>
#include <map>
#include <numeric>
#include <set>
#include <sstream>
#include <tuple>
#include <nntrainer_error.h>
#include <nntrainer_log.h>
#include <profiler.h>
```

Include dependency graph for profiler.cpp:



Namespaces

- nntrainer

9.214.1 Detailed Description

Profiler related codes to be used to benchmark things.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 December 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

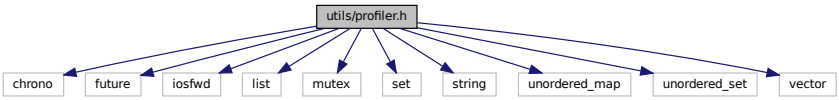
Bug No known bugs except for NYI items

9.215 utils/profiler.h File Reference

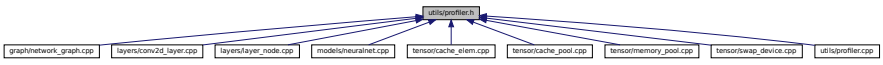
Profiler related codes to be used to benchmark things.

```
#include <chrono>
#include <future>
#include <iosfwd>
#include <list>
#include <mutex>
#include <set>
#include <string>
#include <unordered_map>
#include <unordered_set>
#include <vector>
```

Include dependency graph for profiler.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct nntrainer::profile::ProfileEventData

  Data for each profile event.
- class nntrainer::profile::ProfileListener

  Generic profile listener class to attach to a profiler, this can be inherited to create a custom profile listener.
- class nntrainer::profile::GenericProfileListener

  Generic Profiler Listener.
- class nntrainer::profile::Profiler

  Profiler object.

Namespaces

- nntrainer

Macros

- #define **PROFILE_TIME_START**(event_key)
- #define **PROFILE_TIME_END**(event_key)
- #define **PROFILE_TIME_REGISTER_EVENT**(event_key, event_str)
- #define **PROFILE_MEM_ALLOC**(ptr, size, str)
- #define **PROFILE_MEM_DEALLOC**(ptr)
- #define **PROFILE_CACHE_ALLOC**(ptr, size, str, policy, swap)
- #define **PROFILE_CACHE_DEALLOC**(ptr, policy, swap)
- #define **PROFILE_BEGIN**(listener)
- #define **PROFILE_END**(listener)
- #define **PROFILE_MEM_ANNOTATE**(str)

Typedefs

- using **timepoint** = std::chrono::time_point< std::chrono::steady_clock >

Enumerations

- enum **PROFILE_EVENT** {
  EVENT_TIME_START = 0, **EVENT_TIME_END** = 1, **EVENT_MEM_ALLOC** = 2, **EVENT_MEM_DEALLOC** = 3,
  EVENT_MEM_ANNOTATE = 4 }

Functions

- template<typename T , typename std::enable_if_t< std::is_base_of< ProfileListener, T >::value, T > * = nullptr>
  std::ostream & nntrainer::profile::operator<< (std::ostream &out, T &l)

  Overriding output stream for layers and it's derived class.

9.215.1 Detailed Description

Profiler related codes to be used to benchmark things.

Copyright (C) 2020 Jihoon Lee jhoon.it.lee@samsung.com

Date

09 December 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jihoon Lee jhoon.it.lee@samsung.com

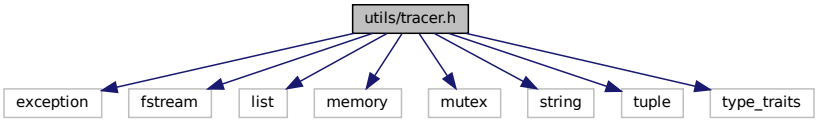
Bug No known bugs except for NYI items

9.216 utils/tracer.h File Reference

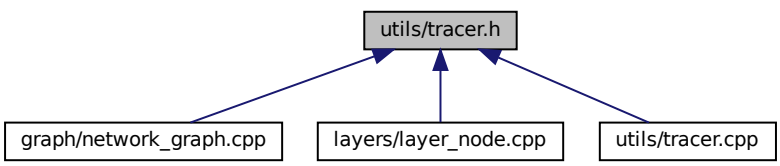
Trace abstract class.

```
#include <exception>
#include <fstream>
#include <list>
#include <memory>
#include <mutex>
#include <string>
#include <tuple>
#include <type_traits>
```

Include dependency graph for tracer.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **TRACE_MEMORY_POINT**(msg)
- #define **TRACE_MEMORY**() std::ostream(nullptr)
- #define **TRACE_TIME_POINT**(msg)
- #define **TRACE_TIME**() std::ostream(nullptr)

9.216.1 Detailed Description

Trace abstract class.

Copyright (C) 2022 Jiho Chu jiho.chu@samsung.com

Date

23 December 2022

See also

https://github.com/nnstreamer/nntrainer

Author

Jiho Chu jiho.chu@samsung.com

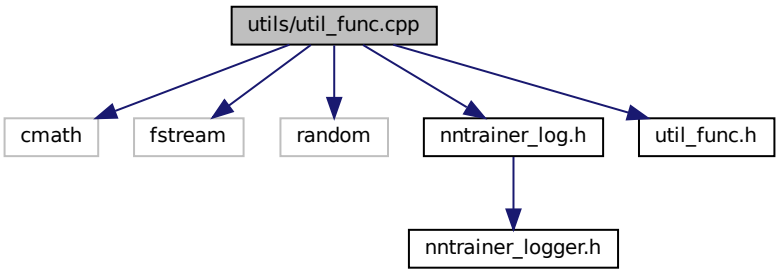
Bug No known bugs except for NYI items

9.217 utils/util_func.cpp File Reference

This is collection of math functions.

```
#include <cmath>
#include <fstream>
#include <random>
#include <nntrainer_log.h>
#include <util_func.h>
```

Include dependency graph for util_func.cpp:



Namespaces

- nntrainer

Functions

- static std::uniform_real_distribution< float > **nntrainer::dist** (-0.5, 0.5)
- unsigned int **nntrainer::getSeed** ()
- float **nntrainer::sqrtFloat** (float x)
- double **nntrainer::sqrtDouble** (double x)
- float **nntrainer::logFloat** (float x)
- float **nntrainer::exp_util** (float x)
- Tensor **nntrainer::rotate_180** (Tensor in)
- bool **nntrainer::isFileExist** (std::string file_name)
- template<typename T >
  static void **nntrainer::checkFile** (const T &file, const char *error_msg)
- void **nntrainer::checkedRead** (std::ifstream &file, char *array, std::streamsize size, const char *error_msg)
- void **nntrainer::checkedWrite** (std::ostream &file, const char *array, std::streamsize size, const char *error_msg)
- std::string **nntrainer::readString** (std::ifstream &file, const char *error_msg)
- void **nntrainer::writeString** (std::ofstream &file, const std::string &str, const char *error_msg)
- bool **nntrainer::endswith** (const std::string &target, const std::string &suffix)
- int **nntrainer::getKeyValue** (const std::string &input_str, std::string &key, std::string &value)
- int **nntrainer::getValues** (int n_str, std::string str, int *value)
- std::vector< std::string > **nntrainer::split** (const std::string &s, const std::regex &reg)
- bool **nntrainer::istrequal** (const std::string &a, const std::string &b)
- char * **nntrainer::getRealpath** (const char *name, char *resolved)
- tm * **nntrainer::getLocaltime** (tm *tp)

Variables

- static auto **nntrainer::rng**

9.217.1 Detailed Description

This is collection of math functions.

Date

08 April 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

9.218 utils/util_func.h File Reference

This is collection of math functions.

9.218.1 Detailed Description

This is collection of math functions.

Date

08 April 2020

See also

https://github.com/nnstreamer/nntrainer

Author

Jijoong Moon jijoong.moon@samsung.com

Bug No known bugs except for NYI items

Index

~ActivationRealizer
nntrainer::ActivationRealizer, 82
~BasicPlanner
nntrainer::BasicPlanner, 104
~BnRealizer
nntrainer::BnRealizer, 123
~CacheElem
nntrainer::CacheElem, 162
~CacheLoader
nntrainer::CacheLoader, 165
~CachePool
nntrainer::CachePool, 170
~CentroidKNN
nntrainer::CentroidKNN, 183
~DataProducer
nntrainer::DataProducer, 201
~DirDataProducer
nntrainer::DirDataProducer, 219
~ErrorNotification
nntrainer::exception::ErrorNotification< Err, >, 236
~Exporter
nntrainer::Exporter, 239
~FlatBufferInterpreter
nntrainer::FlatBufferInterpreter, 250
~FlattenRealizer
nntrainer::FlattenRealizer, 273
~FuncDataProducer
nntrainer::FuncDataProducer, 279
~GenericProfileListener
nntrainer::profile::GenericProfileListener, 284
~GraphRealizer
nntrainer::GraphRealizer, 307
~IniGraphInterpreter
nntrainer::IniGraphInterpreter, 316
~IniSection
nntrainer::IniSection, 322
~InputRealizer
nntrainer::InputRealizer, 355
~IterationQueue
nntrainer::IterationQueue, 369
~LayerNode
nntrainer::LayerNode, 382
~LossRealizer
nntrainer::LossRealizer, 421
~MemoryPlanner
nntrainer::MemoryPlanner, 430
~MemoryPool
nntrainer::MemoryPool, 433
~MultioutRealizer
nntrainer::MultioutRealizer, 448
~ParallelBatch
nntrainer::ParallelBatch, 487
~PluggedLayer
nntrainer::internal::PluggedLayer, 498
~PluggedOptimizer
nntrainer::internal::PluggedOptimizer, 502
~PreprocessL2NormLayer
nntrainer::PreprocessL2NormLayer, 513
~PreviousInputRealizer
nntrainer::PreviousInputRealizer, 517
~ProfileListener
nntrainer::profile::ProfileListener, 520
~RandomDataOneHotProducer
nntrainer::RandomDataOneHotProducer, 548
~RawFileDataProducer
nntrainer::RawFileDataProducer, 556
~RecurrentRealizer
nntrainer::RecurrentRealizer, 566
~RemapRealizer
nntrainer::RemapRealizer, 572
~ScopedView
nntrainer::ScopedView< T >, 618
~SliceRealizer
nntrainer::SliceRealizer, 627
~SwapDevice
nntrainer::SwapDevice, 647
~Task
nntrainer::Task, 653
~TaskAsync
nntrainer::TaskAsync< T >, 657
~TaskExecutor
nntrainer::TaskExecutor, 661
~TfliteInterpreter
nntrainer::TfliteInterpreter, 671

ACT_LEAKY_RELU
nntrainer, 59
ACT_NONE
nntrainer, 59
ACT_RELU
nntrainer, 59
ACT_SIGMOID
nntrainer, 59
ACT_SOFTMAX
nntrainer, 59
ACT_SWISH
nntrainer, 59
ACT_TANH
nntrainer, 59

ACT_UNKNOWN
nntrainer, 59
ActivationType
nntrainer, 59
add_default_object
nntrainer, 64
additional_exec_order
nntrainer::TensorSpecV2, 668
alloc
nntrainer::profile::Profiler, 523
allocate
nntrainer::CachePool, 171
nntrainer::MemoryPool, 433
allowPrecisionLoss
nntrainer::Device, 212
allowSoftPlacement
nntrainer::Device, 213
ALWAYS_SYNCED
nntrainer, 60
annotate
nntrainer::profile::Profiler, 523
app_context.cpp, 739
app_context.h, 741
AppContext::registerFactory< ml::train::LearningRateScheduler >
nntrainer, 65
AppContext::registerFactory< nntrainer::Layer >
nntrainer, 65
AppContext::registerFactory< nntrainer::Optimizer >
nntrainer, 65
applyGradient
nntrainer::internal::PluggedOptimizer, 503
nntrainer::RunOptimizerContext, 610
arg
nntrainer::FlatBufferOpNode, 252
nntrainer::TfOpNode, 674
arity
nntrainer::FlatBufferOpNode, 253
nntrainer::TfOpNode, 675
armcl
nntrainer::Backend, 102
attention_weight
nntrainer, 60
AttentionParams
nntrainer, 59, 60

Backend
nntrainer::Backend, 102
BackendType
nntrainer::Backend, 102
BACKWARD_FUNC_LIFESPAN
nntrainer, 63
base
nntrainer::Backend, 102
BasicRegularizerConstant
nntrainer::props::BasicRegularizerConstant, 109
batch
nntrainer::Iteration, 362
nntrainer::IterationQueue, 370
begin
nntrainer::Iteration, 362, 363
BiasDecay
nntrainer::props::BiasDecay, 112
Bidirectional
nntrainer::props::Bidirectional, 115
blas
nntrainer::Backend, 102
blas_interface.cpp
sgemv_loop, 935
BnRealizer
nntrainer::BnRealizer, 123
BroadcastInfo
nntrainer::Tensor::BroadcastInfo, 125
buffer_axis
nntrainer::Tensor::BroadcastInfo, 125
buffer_size
nntrainer::Tensor::BroadcastInfo, 125
buildTrasposeString
nntrainer, 65

CacheElem
nntrainer::CacheElem, 162
CacheElems
nntrainer::CachePool, 170
CacheLoader
nntrainer::CacheLoader, 165
CachePolicy
nntrainer, 60
CachePool
nntrainer::CachePool, 170
CALC_AGRAD_LIFESPAN
nntrainer, 63
CALC_DERIV_LIFESPAN
nntrainer, 63
CALC_GRAD_DERIV_AGRAD_LIFESPAN
nntrainer, 63
CALC_GRAD_DERIV_LIFESPAN
nntrainer, 63
CALC_GRAD_LIFESPAN
nntrainer, 63
calcDerivative
nntrainer::CentroidKNN, 183
nntrainer::internal::PluggedLayer, 498
nntrainer::LayerNode, 382
nntrainer::PermuteLayer, 493
nntrainer::PreprocessL2NormLayer, 513
calcGradient
nntrainer::internal::PluggedLayer, 498
nntrainer::LayerNode, 382
cancel
nntrainer::TaskExecutor, 662
cancelAsync
nntrainer::CacheLoader, 165
CentroidKNN
nntrainer::CentroidKNN, 183
clean
nntrainer::TaskExecutor, 662
clear

nntrainer::CachePool, 171
nntrainer::MemoryPool, 433
clearOptVar
nntrainer::LayerNode, 382
clip_by_global_norm
nntrainer::WeightSpecV2, 732
clipGradientByGlobalNorm
nntrainer::Weight, 720
clone
nntrainer::Weight, 720
cloneConfiguration
nntrainer::LayerNode, 382
compile
nntrainer::GraphCompiler, 288
compiler/activation_realizer.cpp, 743
compiler/activation_realizer.h, 744
compiler/bn_realizer.cpp, 746
compiler/bn_realizer.h, 747
compiler/compiler.h, 748
compiler/compiler_fwd.h, 749
compiler/flatbuffer_interpreter.cpp, 750
compiler/flatbuffer_interpreter.h, 751
compiler/flatbuffer_opnode.h, 752
compiler/flatten_realizer.cpp, 754
compiler/flatten_realizer.h, 755
compiler/ini_interpreter.cpp, 756
compiler/ini_interpreter.h, 757
compiler/input_realizer.cpp, 759
compiler/input_realizer.h, 760
compiler/interpreter.h, 761
compiler/loss_realizer.h, 763
compiler/multiout_realizer.h, 765
compiler/previous_input_realizer.cpp, 767
compiler/previous_input_realizer.h, 768
compiler/realizer.h, 769
compiler/recurrent_realizer.h, 770
compiler/remap_realizer.h, 772
compiler/slice_realizer.cpp, 774
compiler/slice_realizer.h, 775
compiler/tflite_interpreter.cpp, 776
compiler/tflite_interpreter.h, 777
compiler/tflite_opnode.cpp, 778
compiler/tflite_opnode.h, 779
CompleteCallback
nntrainer::TaskExecutor, 661
compute
nntrainer::props::Padding1D, 479
nntrainer::props::Padding2D, 482
computeSpace
nntrainer, 66
configureRunContext
nntrainer::LayerNode, 383
Connection
nntrainer::Connection, 192, 193
cpu
nntrainer::Device, 212
CREATE_IF_EMPTY_DIMS
tensor.cpp, 963
createfunc
nntrainer, 76
createLayerNode
nntrainer, 66, 67
createObject
nntrainer::AppContext, 87

dataset/data_iteration.cpp, 781
dataset/data_iteration.h, 782
dataset/data_producer.h, 783
dataset/databuffer.cpp, 784
dataset/databuffer.h, 785
dataset/databuffer_factory.cpp, 786
dataset/databuffer_factory.h, 787
dataset/dir_data_producers.h, 788
dataset/func_data_producer.cpp, 790
dataset/func_data_producer.h, 791
dataset/iteration_queue.cpp, 792
dataset/iteration_queue.h, 793
dataset/random_data_producers.cpp, 795
dataset/random_data_producers.h, 796
dataset/raw_file_data_producer.cpp, 797
dataset/raw_file_data_producer.h, 798
dealloc
nntrainer::profile::Profiler, 524
deallocate
nntrainer::CachePool, 171
nntrainer::MemoryPool, 434
decay
nntrainer::WeightSpecV2, 733
decompile
nntrainer::GraphCompiler, 289
delegate.h, 800
DelegateConfig
nntrainer::DelegateConfig, 208
deserialize
nntrainer::FlatBufferInterpreter, 250
nntrainer::GraphInterpreter, 290
nntrainer::IniGraphInterpreter, 317
nntrainer::TfliteInterpreter, 671
deserialize_v1
nntrainer::GraphInterpreter, 290
Device
nntrainer::Device, 212
DeviceType
nntrainer::Device, 212
Dilation
nntrainer::props::Dilation, 216
dim
nntrainer::TensorSpecV2, 668
DirDataProducer
nntrainer::DirDataProducer, 218
DirPath
nntrainer::props::DirPath, 222
DisableBias
nntrainer::props::DisableBias, 226
done
nntrainer::Task, 654
DropOutRate

nntrainer::props::DropOutRate, 229
DynamicTimeSequence
nntrainer::props::DynamicTimeSequence, 231

end
nntrainer::Iteration, 363, 364
nntrainer::MemoryRequest, 439
nntrainer::profile::Profiler, 524
EnumList
nntrainer::props::ActivationTypeInfo, 83
nntrainer::props::FlipDirectionInfo, 275
nntrainer::props::InitializerInfo, 330
nntrainer::props::PoolingTypeInfo, 508
nntrainer::props::RegularizerInfo, 570
nntrainer::props::ReturnAttentionWeightInfo, 580
EnumStr
nntrainer::props::ActivationTypeInfo, 83
nntrainer::props::FlipDirectionInfo, 275
nntrainer::props::InitializerInfo, 330
nntrainer::props::PoolingTypeInfo, 508
EPOCH_LIFESPAN
nntrainer, 64
Epsilon
nntrainer::props::Epsilon, 234
erase_ini
nntrainer::IniWrapper, 343
ErrorNotification
nntrainer::exception::ErrorNotification< Err, >, 236
ESTRPIPE
nntrainer_error.h, 908
executeInPlace
nntrainer::InitLayerContext, 332
nntrainer::LayerNode, 383
nntrainer::RunLayerContext, 587
ExecutionMode
nntrainer, 60
ExecutionOrder
nntrainer::GraphNode, 293
Exporter
nntrainer::Exporter, 239
exportTo
nntrainer::CentroidKNN, 184
nntrainer::DataProducer, 201
nntrainer::FuncDataProducer, 279
nntrainer::internal::PluggedLayer, 498
nntrainer::LayerNode, 384
nntrainer::PermuteLayer, 494
nntrainer::PreprocessL2NormLayer, 514
nntrainer::RawFileDataProducer, 556
External, 241

file_size
nntrainer::props::FilePath, 245
FilePath
nntrainer::props::FilePath, 244
finalize
nntrainer::CentroidKNN, 184
nntrainer::DataProducer, 201
nntrainer::DirDataProducer, 219
nntrainer::FuncDataProducer, 280
nntrainer::internal::PluggedLayer, 499
nntrainer::LayerNode, 384
nntrainer::PermuteLayer, 494
nntrainer::PreprocessL2NormLayer, 514
nntrainer::RandomDataOneHotProducer, 549
nntrainer::RawFileDataProducer, 556
nntrainer::TfOpNode, 675
fini_global_context_nntrainer
nntrainer, 68
finish
nntrainer::CacheLoader, 166
nntrainer::SwapDevice, 647
FIRST_ACCESS
nntrainer::CacheElem, 162
FIRST_LAST_SKIP
nntrainer, 60
FlatBufferInterpreter
nntrainer::FlatBufferInterpreter, 250
FlatBufferOpNode
nntrainer::FlatBufferOpNode, 252
flush
nntrainer::CachePool, 172
flushExcept
nntrainer::CachePool, 173
FORWARD_DERIV_LIFESPAN
nntrainer, 63
FORWARD_FUNC_LIFESPAN
nntrainer, 63
FORWARD_GRAD_AGRAD_LIFESPAN
nntrainer, 63
FORWARD_GRAD_LIFESPAN
nntrainer, 63
forwarding
nntrainer::CentroidKNN, 184
nntrainer::internal::PluggedLayer, 499
nntrainer::LayerNode, 385
nntrainer::PermuteLayer, 494
nntrainer::PreprocessL2NormLayer, 514
FromExportable
nntrainer::IniSection, 322
FuncDataProducer
nntrainer::FuncDataProducer, 279

Generator
nntrainer::DataProducer, 200
GenericProfileListener
nntrainer::profile::GenericProfileListener, 283
get
nntrainer::ScopedView< T >, 618
getActivationToBeRealized
nntrainer::LayerNode, 386
getActivationType
nntrainer::LayerNode, 386
getBuffer
nntrainer::SwapDevice, 648
getBuiltinOps
nntrainer::FlatBufferOpNode, 253
nntrainer::TfOpNode, 676

getCachePolicy
nntrainer::CachePool, 174
getData
nntrainer::Task, 654
getDefaultLearningRate
nntrainer::internal::PluggedOptimizer, 503
getDevice
nntrainer::Device, 213
getDevicePath
nntrainer::SwapDevice, 648
getDim
nntrainer::Var_Grad, 701
getDistribute
nntrainer::LayerNode, 387
getExecutionOrder
nntrainer::GraphNode, 293
nntrainer::LayerNode, 387
getFlatten
nntrainer::LayerNode, 388
getGradient
nntrainer::RunOptimizerContext, 611
nntrainer::Var_Grad, 701
getGradientName
nntrainer::Var_Grad, 701
getGradientNorm
nntrainer::Var_Grad, 701
getGradientRef
nntrainer::Var_Grad, 702
getId
nntrainer::CacheElem, 163
getIncomingDerivative
nntrainer::RunLayerContext, 587
getIndex
nntrainer::Connection, 193
getIniName
nntrainer::IniWrapper, 343
getInput
nntrainer::LayerNode, 388
nntrainer::RunLayerContext, 588, 589
getInputConnectionIndex
nntrainer::LayerNode, 388
getInputConnectionName
nntrainer::LayerNode, 389
getInputConnections
nntrainer::GraphNode, 293
nntrainer::LayerNode, 389
getInputDimensions
nntrainer::InitLayerContext, 333
nntrainer::LayerNode, 389
getInputGrad
nntrainer::LayerNode, 390
nntrainer::RunLayerContext, 589
getInputNodes
nntrainer::FlatBufferOpNode, 254
nntrainer::TfOpNode, 676
getInputs
nntrainer::FlatBufferOpNode, 254
nntrainer::TfOpNode, 676
getInputsRef
nntrainer::Iteration, 364
nntrainer::Sample, 615
getIteration
nntrainer::RunOptimizerContext, 611
getLabel
nntrainer::RunLayerContext, 590
getLabelsRef
nntrainer::Iteration, 365
nntrainer::Sample, 615
getLearningRate
nntrainer::RunOptimizerContext, 612
getLength
nntrainer::CacheElem, 163
getLoss
nntrainer::LayerNode, 390
nntrainer::RunLayerContext, 590
getMemory
nntrainer::CachePool, 174
nntrainer::MemoryPool, 434
getName
nntrainer::CachePool, 175
nntrainer::Connection, 194
nntrainer::GraphNode, 294
nntrainer::IniSection, 323
nntrainer::InitLayerContext, 333
nntrainer::IniWrapper, 343
nntrainer::LayerNode, 390
nntrainer::RunLayerContext, 591
nntrainer::Var_Grad, 702
getNumInputConnections
nntrainer::LayerNode, 391
getNumInputs
nntrainer::InitLayerContext, 333
nntrainer::LayerNode, 392
nntrainer::RunLayerContext, 591
getNumOptVariable
nntrainer::Weight, 720
getNumOutputConnections
nntrainer::LayerNode, 392
getNumOutputs
nntrainer::LayerNode, 393
nntrainer::RunLayerContext, 591
getNumRequestedOutputs
nntrainer::InitLayerContext, 334
getNumTensors
nntrainer::InitLayerContext, 334
nntrainer::RunLayerContext, 592
getNumWeightOptVar
nntrainer::RunLayerContext, 592
getNumWeights
nntrainer::InitLayerContext, 334
nntrainer::LayerNode, 393
nntrainer::RunLayerContext, 593
getNumWorkers
nntrainer::ParallelBatch, 487
getOptimizerVariable
nntrainer::RunOptimizerContext, 612

getOptimizerVariableDim
nntrainer::internal::PluggedOptimizer, 503
getOptimizerVariableRef
nntrainer::Weight, 720
getOptionType
nntrainer::FlatBufferOpNode, 254
nntrainer::TfOpNode, 677
getOpType
nntrainer::FlatBufferOpNode, 255
nntrainer::TfOpNode, 677
getOutgoingDerivative
nntrainer::RunLayerContext, 593
getOutput
nntrainer::LayerNode, 393
nntrainer::RunLayerContext, 594, 595
getOutputConnection
nntrainer::LayerNode, 394
getOutputConnections
nntrainer::GraphNode, 294
nntrainer::LayerNode, 394
getOutputDimensions
nntrainer::LayerNode, 395
getOutputGrad
nntrainer::LayerNode, 395
nntrainer::RunLayerContext, 595
getOutputGradUnsafe
nntrainer::LayerNode, 396
nntrainer::RunLayerContext, 596
getOutputs
nntrainer::FlatBufferOpNode, 255
nntrainer::TfOpNode, 677
getOutSpecs
nntrainer::InitLayerContext, 334
getPriority
nntrainer::TaskAsync< T >, 657
getProperties
nntrainer::AppContext, 88
getRegularizationLoss
nntrainer::RunLayerContext, 597
getResult
nntrainer::Exporter, 239
getRunContext
nntrainer::LayerNode, 396, 397
getSharedFrom
nntrainer::LayerNode, 397
getTensor
nntrainer::RunLayerContext, 598
getTensorGrad
nntrainer::RunLayerContext, 599
getTensorName
nntrainer::RunLayerContext, 599
getTensorsSpec
nntrainer::InitLayerContext, 335
getTimeout
nntrainer::TaskAsync< T >, 658
getTrainable
nntrainer::GraphNode, 294
nntrainer::LayerNode, 397
nntrainer::RunLayerContext, 600
getType
nntrainer::BasicPlanner, 105
nntrainer::CentroidKNN, 185
nntrainer::DataProducer, 202
nntrainer::DirDataProducer, 219
nntrainer::FuncDataProducer, 280
nntrainer::GraphNode, 294
nntrainer::internal::PluggedLayer, 499
nntrainer::internal::PluggedOptimizer, 504
nntrainer::LayerNode, 398
nntrainer::MemoryPlanner, 430
nntrainer::OptimizedV1Planner, 465
nntrainer::OptimizedV2Planner, 468
nntrainer::OptimizedV3Planner, 470
nntrainer::PermuteLayer, 495
nntrainer::PreprocessL2NormLayer, 515
nntrainer::RandomDataOneHotProducer, 549
nntrainer::RawFileDataProducer, 557
nntrainer::Task, 654
nntrainer::TaskAsync< T >, 658
getVariable
nntrainer::Var_Grad, 703
getVariableRef
nntrainer::Var_Grad, 703, 704
getWeight
nntrainer::LayerNode, 398
nntrainer::RunLayerContext, 600
nntrainer::RunOptimizerContext, 613
getWeightGrad
nntrainer::LayerNode, 399
nntrainer::RunLayerContext, 601
getWeightName
nntrainer::LayerNode, 399
nntrainer::RunLayerContext, 601
getWeightObject
nntrainer::LayerNode, 399
nntrainer::RunLayerContext, 602
getWeightOptVar
nntrainer::RunLayerContext, 602
getWeights
nntrainer::FlatBufferOpNode, 255, 256
nntrainer::LayerNode, 400
nntrainer::TfOpNode, 678
getWeightsSpec
nntrainer::InitLayerContext, 335
getWeightWrapper
nntrainer::LayerNode, 401
getWork
nntrainer::Task, 654
getWorkingPath
nntrainer::AppContext, 88
Global
nntrainer::AppContext, 88
nntrainer::profile::Profiler, 525
gpu
nntrainer::Device, 212
grad

nntrainer::Var_Grad, 708
gradient_spec
nntrainer::VarGradSpecV2, 711
graph/connection.cpp, 801
graph/connection.h, 802
graph/graph_node.h, 803
graph/network_graph.h, 804
GraphNodeIterator
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 298
GraphNodeReverseIterator
nntrainer::GraphNodeReverseIterator< T_iterator >, 305
grucell_calcGradient
nntrainer, 68
grucell_forwarding
nntrainer, 68

handleWork
nntrainer::TaskExecutor, 662
hasGradient
nntrainer::Var_Grad, 704
hasInputShapeProperty
nntrainer::LayerNode, 401
hasWorkingDirectory
nntrainer::AppContext, 89
HiddenStateActivation
nntrainer::props::HiddenStateActivation, 312

IndexType
nntrainer::AppContext, 86
INFERENCE
nntrainer, 62
IniGraphInterpreter
nntrainer::IniGraphInterpreter, 316
IniSection
nntrainer::IniSection, 319–321
initCacheElemIter
nntrainer::CachePool, 175
initExecIdsIter
nntrainer::CachePool, 175
initializeGradient
nntrainer::Var_Grad, 705
initializer
nntrainer::TensorSpecV2, 668
initializeVariable
nntrainer::Var_Grad, 705
InitLayerContext
nntrainer::InitLayerContext, 332
IniWrapper
nntrainer::IniWrapper, 342
INOUT_INDEX
nntrainer, 62
InPlace
nntrainer, 62
InputConnection
nntrainer::props::InputConnection, 352
inputHasGradient
nntrainer::RunLayerContext, 603
InputRealizer
nntrainer::InputRealizer, 355
IntegrateBias
nntrainer::props::IntegrateBias, 360
IntIndexType
nntrainer::AppContext, 86
invalidate
nntrainer::CachePool, 176
is_dependent
nntrainer::Var_Grad, 708
is_first_access_gradient
nntrainer::Var_Grad, 708
is_last_access_gradient
nntrainer::Var_Grad, 709
isActive
nntrainer::CacheElem, 163
isAllocated
nntrainer::CachePool, 176
nntrainer::MemoryPool, 435
isDependent
nntrainer::Var_Grad, 706
isEmpty
nntrainer::ScopedView< T >, 618
nntrainer::ViewQueue< T >, 713
isFinalized
nntrainer::LayerNode, 401
isGradientClipByGlobalNorm
nntrainer::RunLayerContext, 603
nntrainer::Weight, 721
isGradientFirstAccess
nntrainer::RunLayerContext, 604
nntrainer::Var_Grad, 706
isGradientLastAccess
nntrainer::RunLayerContext, 604
nntrainer::Var_Grad, 706
isInputNode
nntrainer::FlatBufferOpNode, 256
nntrainer::TfOpNode, 678
isLabelAvailable
nntrainer::RunLayerContext, 605
isLastCacheElemIter
nntrainer::CachePool, 176
isLastExecIdsIter
nntrainer::CachePool, 177
isMultiThreadSafe
nntrainer::DataProducer, 202
nntrainer::DirDataProducer, 219
nntrainer::RandomDataOneHotProducer, 549
isNeedReorder
nntrainer::TfOpNode, 679
isOperating
nntrainer::SwapDevice, 648
isOutputNode
nntrainer::FlatBufferOpNode, 256
nntrainer::TfOpNode, 679
isTrainable
nntrainer::TfOpNode, 679
isValid

nntrainer::props::Axis, 100
nntrainer::props::BasicRegularizer, 107
nntrainer::props::BasicRegularizerConstant, 109
nntrainer::props::DirPath, 223
nntrainer::props::DropOutRate, 229
nntrainer::props::Epsilon, 234
nntrainer::props::FilePath, 245
nntrainer::props::Loss, 419
nntrainer::props::Momentum, 445
nntrainer::props::Name, 451
nntrainer::props::NumClass, 459
nntrainer::props::Padding1D, 480
nntrainer::props::Padding2D, 483
nntrainer::props::PermuteDims, 491
nntrainer::props::SplitDimension, 634
nntrainer::PropsNNSModelPath, 538
nntrainer::PropsTflModelPath, 543
isVirtualNode
nntrainer::FlatBufferOpNode, 256
nntrainer::TfOpNode, 679
isWeightDecay
nntrainer::Weight, 722
isWeightDependent
nntrainer::RunLayerContext, 605
isWeightRegularizerL2Norm
nntrainer::Weight, 722
iterate_prop
nntrainer, 69, 70
Iteration
nntrainer::Iteration, 361
ITERATION_CONSIST
nntrainer, 60
ITERATION_LIFESPAN
nntrainer, 63
IterationQueue
nntrainer::IterationQueue, 369

key
nntrainer::props::AverageAttentionWeight, 98
nntrainer::props::Axis, 101
nntrainer::props::BasicRegularizerConstant, 110
nntrainer::props::BiasDecay, 113
nntrainer::props::ClipGradByGlobalNorm, 188
nntrainer::props::DecayRate, 205
nntrainer::props::DecaySteps, 207
nntrainer::props::Dilation, 216
nntrainer::props::DirPath, 224
nntrainer::props::DropOutRate, 230
nntrainer::props::Epsilon, 234
nntrainer::props::FilePath, 246
nntrainer::props::FilterSize, 248
nntrainer::props::Flatten, 271
nntrainer::props::GenericShape, 287
nntrainer::props::InDim, 314
nntrainer::props::InputConnection, 352
nntrainer::props::InputShape, 358
nntrainer::props::Iteration, 367
nntrainer::props::KernelSize, 373
nntrainer::props::LearningRate, 412
nntrainer::props::Loss, 420
nntrainer::props::MaxTimestep, 427
nntrainer::props::MoL_K, 443
nntrainer::props::Momentum, 446
nntrainer::props::Name, 452
nntrainer::props::NumClass, 459
nntrainer::props::NumHeads, 461
nntrainer::props::OutDim, 473
nntrainer::props::OutputShape, 476
nntrainer::props::Padding1D, 480
nntrainer::props::Padding2D, 483
nntrainer::props::PermuteDims, 491
nntrainer::props::PoolSize, 511
nntrainer::props::ProjectedKeyDim, 528
nntrainer::props::ProjectedValueDim, 530
nntrainer::props::RandomTranslate, 553
nntrainer::props::ResetAfter, 575
nntrainer::props::ReturnAttentionWeight, 579
nntrainer::props::SharedFrom, 623
nntrainer::props::SplitNumber, 636
nntrainer::props::Stride, 643
nntrainer::props::TargetShape, 652
nntrainer::props::Timestep, 686
nntrainer::props::Unit, 693
nntrainer::props::WeightDecay, 726
nntrainer::props::WeightRegularizerConstant, 731
nntrainer::props::ZeroIdxMask, 737
nntrainer::PropsBufferSize, 532
nntrainer::PropsMax, 534
nntrainer::PropsMin, 536
nntrainer::PropsNNSModelPath, 538
nntrainer::PropsNumSamples, 541
nntrainer::PropsTflModelPath, 543

L2NORM
nntrainer, 64
LabelLayer
nntrainer::props::LabelLayer, 375
LAST_ACCESS
nntrainer::CacheElem, 162
Layer, 376, 377
LayerNode
nntrainer::LayerNode, 381
layers/acti_func.cpp, 806
layers/activation_layer.cpp, 807
layers/activation_layer.h, 808
layers/addition_layer.cpp, 809
layers/addition_layer.h, 810
layers/attention_layer.cpp, 811
layers/attention_layer.h, 812
layers/bn_layer.cpp, 813
layers/bn_layer.h, 814
layers/centroid_knn.cpp, 815
layers/centroid_knn.h, 816
layers/common_properties.cpp, 817
layers/common_properties.h, 818
layers/concat_layer.cpp, 823
layers/concat_layer.h, 824
layers/conv1d_layer.h, 824

layers/conv2d_layer.h, 825
layers/dropout.h, 826
layers/embedding.cpp, 827
layers/embedding.h, 828
layers/fc_layer.cpp, 829
layers/fc_layer.h, 830
layers/flatten_layer.cpp, 831
layers/flatten_layer.h, 832
layers/gru.cpp, 833
layers/gru.h, 834
layers/grucell.cpp, 835
layers/grucell.h, 836
layers/input_layer.cpp, 837
layers/input_layer.h, 838
layers/layer_context.cpp, 839
layers/layer_context.h, 840
layers/layer_devel.h, 841
layers/layer_impl.h, 842
layers/layer_node.cpp, 844
layers/layer_node.h, 846
layers/layer_normalization_layer.cpp, 848
layers/layer_normalization_layer.h, 849
layers/loss/constant_derivative_loss_layer.cpp, 849
layers/loss/cross_entropy_loss_layer.h, 850
layers/loss/cross_entropy_sigmoid_loss_layer.cpp, 851
layers/loss/cross_entropy_sigmoid_loss_layer.h, 852
layers/loss/kld_loss_layer.cpp, 853
layers/loss/kld_loss_layer.h, 854
layers/loss/loss_layer.cpp, 855
layers/loss/loss_layer.h, 856
layers/loss/mse_loss_layer.cpp, 857
layers/loss/mse_loss_layer.h, 858
layers/lstm.cpp, 858
layers/lstm.h, 859
layers/lstmcell.cpp, 860
layers/lstmcell.h, 861
layers/lstmcell_core.cpp, 862
layers/lstmcell_core.h, 863
layers/mol_attention_layer.cpp, 864
layers/mol_attention_layer.h, 865
layers/multi_head_attention_layer.cpp, 866
layers/multi_head_attention_layer.h, 867
layers/multiout_layer.cpp, 868
layers/multiout_layer.h, 869
layers/nnstreamer_layer.cpp, 869
layers/nnstreamer_layer.h, 871
layers/permute_layer.cpp, 872
layers/permute_layer.h, 873
layers/plugged_layer.h, 874
layers/pooling2d_layer.cpp, 875
layers/pooling2d_layer.h, 876
layers/positional_encoding_layer.cpp, 877
layers/positional_encoding_layer.h, 878
layers/preprocess_flip_layer.cpp, 879
layers/preprocess_flip_layer.h, 880
layers/preprocess_l2norm_layer.cpp, 881
layers/preprocess_l2norm_layer.h, 882
layers/reduce_mean_layer.cpp, 883
layers/reduce_mean_layer.h, 884
layers/reshape_layer.h, 885
layers/rnn.cpp, 886
layers/rnn.h, 887
layers/rnncell.cpp, 887
layers/rnncell.h, 889
layers/split_layer.cpp, 889
layers/split_layer.h, 890
layers/tflite_layer.cpp, 891
layers/tflite_layer.h, 892
layers/time_dist.cpp, 893
layers/time_dist.h, 894
layers/zoneout_lstmcell.cpp, 895
layers/zoneout_lstmcell.h, 896
load
    nntrainer::CacheLoader, 166
loadAsync
    nntrainer::CacheLoader, 166, 167
loadExec
    nntrainer::CachePool, 177
loadExecOnce
    nntrainer::CachePool, 178
loadProperties
    nntrainer, 71
loc
    nntrainer::MemoryRequest, 439
Loss
    nntrainer::props::Loss, 418
LossRealizer
    nntrainer::LossRealizer, 421
ls
    nntrainer::TensorSpecV2, 668

MAX_LIFESPAN
    nntrainer, 64
MAYBE_MODIFYING_VIEW
    nntrainer::TensorSpecV2, 667
MemoryData
    nntrainer::MemoryData< T >, 428, 429
MemoryPool
    nntrainer::MemoryPool, 432
MemoryRequest
    nntrainer::MemoryRequest, 438, 439
minMemoryRequirement
    nntrainer::MemoryPool, 435
ml_error_e
    nntrainer_error.h, 908
ML_ERROR_INVALID_PARAMETER
    nntrainer_error.h, 909
ML_ERROR_NONE
    nntrainer_error.h, 909
ML_ERROR_NOT_SUPPORTED
    nntrainer_error.h, 909
ML_ERROR_OUT_OF_MEMORY
    nntrainer_error.h, 909
ML_ERROR_PERMISSION_DENIED
    nntrainer_error.h, 909
ML_ERROR_TIMED_OUT
    nntrainer_error.h, 909

ML_ERROR_TRY_AGAIN
nntrainer_error.h, 909
ML_ERROR_UNKNOWN
nntrainer_error.h, 909
ml_logd
nntrainer_log.h, 910
ml_loge
nntrainer_log.h, 910
ml_logi
nntrainer_log.h, 911
ml_logw
nntrainer_log.h, 911
model_loader.cpp
NN_INI_RETURN_STATUS, 902
models/dynamic_training_optimization.cpp, 896
models/dynamic_training_optimization.h, 898
models/execution_mode.h, 899
models/model_common_properties.cpp, 899
models/model_common_properties.h, 900
models/model_loader.cpp, 901
models/model_loader.h, 903
models/neuralnet.cpp, 904
models/neuralnet.h, 905
MoLAttentionParams
nntrainer, 62
Momentum
nntrainer::props::Momentum, 445

Name
nntrainer::props::Name, 450
name
nntrainer::TensorSpecV2, 668
needsCalcDerivative
nntrainer::LayerNode, 402
needsCalcGradient
nntrainer::LayerNode, 402, 403
NN_INI_RETURN_STATUS
model_loader.cpp, 902
NNTR_THROW_IF
nntrainer_error.h, 908
NNTR_THROW_IF_CLEANUP
nntrainer_error.h, 908
nntrainer, 49
ACT_LEAKY_RELU, 59
ACT_NONE, 59
ACT_RELU, 59
ACT_SIGMOID, 59
ACT_SOFTMAX, 59
ACT_SWISH, 59
ACT_TANH, 59
ACT_UNKNOWN, 59
ActivationType, 59
add_default_object, 64
ALWAYS_SYNCED, 60
AppContext::registerFactory< ml::train::LearningRateScheduler >, 65
AppContext::registerFactory< nntrainer::Layer >, 65
AppContext::registerFactory< nntrainer::Optimizer >, 65
attention_weight, 60
AttentionParams, 59, 60
BACKWARD_FUNC_LIFESPAN, 63
buildTrasposeString, 65
CachePolicy, 60
CALC_AGRAD_LIFESPAN, 63
CALC_DERIV_LIFESPAN, 63
CALC_GRAD_DERIV_AGRAD_LIFESPAN, 63
CALC_GRAD_DERIV_LIFESPAN, 63
CALC_GRAD_LIFESPAN, 63
computeSpace, 66
createfunc, 76
createLayerNode, 66, 67
EPOCH_LIFESPAN, 64
ExecutionMode, 60
fini_global_context_nntrainer, 68
FIRST_LAST_SKIP, 60
FORWARD_DERIV_LIFESPAN, 63
FORWARD_FUNC_LIFESPAN, 63
FORWARD_GRAD_AGRAD_LIFESPAN, 63
FORWARD_GRAD_LIFESPAN, 63
grucell_calcGradient, 68
grucell_forwarding, 68
INFERENCE, 62
INOUT_INDEX, 62
InPlace, 62
iterate_prop, 69, 70
ITERATION_CONSIST, 60
ITERATION_LIFESPAN, 63
L2NORM, 64
loadProperties, 71
MAX_LIFESPAN, 64
MoLAttentionParams, 62
NO_READ_CONSIST, 60
NO_WRITE_BACK, 60
NON_RESTRICTING, 62
NONE, 62, 64
OUTPUT, 62
overlap, 72
parse_properties, 72
PRINT_INST_INFO, 63
PRINT_METRIC, 63
PRINT_PROP, 63
PRINT_PROP_META, 63
PRINT_SHAPE_INFO, 63
PRINT_WEIGHTS, 63
PrintOption, 63
propagateTimestep, 73
QUERY, 62
READ_CONSIST, 60
RESTRICTING, 62
rng, 77
setInPlaceSharedMemoryConfigByLayer, 74
suffixSpec, 74
TEMPORAL, 60
TensorLifespan, 63

UNKNOWN, 64
UNMANAGED, 63
VALIDATE, 62
validateIntervalOverlap, 75
value, 60, 62
VarGradSpec, 58
WeightRegularizer, 64
WeightSpec, 58
WRITE_BACK, 60
nntrainer::ActivationRealizer, 81
~ActivationRealizer, 82
realize, 82
nntrainer::AppContext, 85
createObject, 87
getProperties, 88
getWorkingPath, 88
Global, 88
hasWorkingDirectory, 89
IndexType, 86
IntIndexType, 86
registerFactory, 89, 90
registerLayer, 91
registerOptimizer, 91
registerPluggableFromDirectory, 92
setWorkingDirectory, 92
unknownFactory, 94
unsetWorkingDirectory, 94
nntrainer::Backend, 101
armcl, 102
Backend, 102
BackendType, 102
base, 102
blas, 102
setNumThreads, 103
nntrainer::BasicPlanner, 103
~BasicPlanner, 104
getType, 105
planLayout, 105
nntrainer::BnRealizer, 122
~BnRealizer, 123
BnRealizer, 123
realize, 124
nntrainer::CacheElem, 161
~CacheElem, 162
CacheElem, 162
FIRST_ACCESS, 162
getId, 163
getLength, 163
isActive, 163
LAST_ACCESS, 162
NONE, 162
Options, 162
reset, 163
swapIn, 163
swapOut, 164
nntrainer::CacheLoader, 164
~CacheLoader, 165
CacheLoader, 165
cancelAsync, 165
finish, 166
load, 166
loadAsync, 166, 167
nntrainer::CachePool, 168
~CachePool, 170
allocate, 171
CacheElems, 170
CachePool, 170
clear, 171
deallocate, 171
flush, 172
flushExcept, 173
getCachePolicy, 174
getMemory, 174
getName, 175
initCacheElemIter, 175
initExecIdsIter, 175
invalidate, 176
isAllocated, 176
isLastCacheElemIter, 176
isLastExecIdsIter, 177
loadExec, 177
loadExecOnce, 178
requestMemory, 178
unloadExec, 179
validate, 179
nntrainer::CentroidKNN, 182
~CentroidKNN, 183
calcDerivative, 183
CentroidKNN, 183
exportTo, 184
finalize, 184
forwarding, 184
getType, 185
requireLabel, 185
setProperty, 185
supportBackwarding, 186
nntrainer::Connection, 191
Connection, 192, 193
getIndex, 193
getName, 194
operator=, 194, 195
operator==, 195
toString, 196
nntrainer::DataProducer, 199
~DataProducer, 201
exportTo, 201
finalize, 201
Generator, 200
getType, 202
isMultiThreadSafe, 202
setProperty, 203
size, 203
SIZE_UNDEFINED, 204
nntrainer::DelegateConfig, 208
DelegateConfig, 208
nntrainer::Device, 211

allowPrecisionLoss, 212
allowSoftPlacement, 213
cpu, 212
Device, 212
DeviceType, 212
getDevice, 213
gpu, 212
npu, 212
setDevice, 213
setMemoryFraction, 214
nntrainer::DirDataProducer, 217
~DirDataProducer, 219
DirDataProducer, 218
finalize, 219
getType, 219
isMultiThreadSafe, 219
setProperty, 220
size, 220
nntrainer::exception, 77
nntrainer::exception::ErrorNotification< Err, >, 235
~ErrorNotification, 236
ErrorNotification, 236
operator<<, 237
nntrainer::exception::not_supported, 456
nntrainer::exception::permission_denied, 488
nntrainer::Exporter, 238
~Exporter, 239
Exporter, 239
getResult, 239
saveResult, 240
nntrainer::FlatBufferInterpreter, 249
~FlatBufferInterpreter, 250
deserialize, 250
FlatBufferInterpreter, 250
serialize, 250
nntrainer::FlatBufferOpNode, 251
arg, 252
arity, 253
FlatBufferOpNode, 252
getBuiltinOps, 253
getInputNodes, 254
getInputs, 254
getOptionType, 254
getOpType, 255
getOutputs, 255
getWeights, 255, 256
isInputNode, 256
isOutputNode, 256
isVirtualNode, 256
setArg, 257
setBuiltinOptions, 257
setLayerNode, 257
setOpType, 258
nntrainer::FlattenRealizer, 272
~FlattenRealizer, 273
realize, 273
nntrainer::FuncDataProducer, 278
~FuncDataProducer, 279
exportTo, 279
finalize, 280
FuncDataProducer, 279
getType, 280
setProperty, 280
nntrainer::GraphCompiler, 288
compile, 288
decompile, 289
nntrainer::GraphInterpreter, 289
deserialize, 290
deserialize_v1, 290
serialize, 291
serialize_v1, 291
nntrainer::GraphNode, 292
ExecutionOrder, 293
getExecutionOrder, 293
getInputConnections, 293
getName, 294
getOutputConnections, 294
getTrainable, 294
getType, 294
setExecutionOrder, 295
setName, 295
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 296
GraphNodeIterator, 298
operator!=, 302
operator*, 298
operator+, 298
operator++, 299
operator+=, 299
operator-, 300
operator->, 302
operator--, 301
operator-=, 301
operator==, 302
value_type, 303
nntrainer::GraphNodeReverseIterator< T_iterator >, 304
GraphNodeReverseIterator, 305
operator*, 305
operator->, 306
nntrainer::GraphRealizer, 306
~GraphRealizer, 307
realize, 308
nntrainer::IniGraphInterpreter, 315
~IniGraphInterpreter, 316
deserialize, 317
IniGraphInterpreter, 316
serialize, 317
nntrainer::IniSection, 318
~IniSection, 322
FromExportable, 322
getName, 323
IniSection, 319–321
operator!=, 323
operator<<, 328
operator+, 324, 325

operator+=, 325, 326
operator==, 327
print, 327
setEntry, 328
nntrainer::InitLayerContext, 330
executeInPlace, 332
getInputDimensions, 333
getName, 333
getNumInputs, 333
getNumRequestedOutputs, 334
getNumTensors, 334
getNumWeights, 334
getOutSpecs, 334
getTensorsSpec, 335
getWeightsSpec, 335
InitLayerContext, 332
outSpec, 335
requestOutputs, 336
requestTensor, 336, 337
requestWeight, 337, 338
setDynDimFlagInputDimension, 339
setEffDimFlagInputDimension, 339
setOutputDimensions, 339
TensorSpec, 332
validate, 340
nntrainer::IniWrapper, 341
erase_ini, 343
getIniName, 343
getName, 343
IniWrapper, 342
operator!=, 344
operator<<, 350
operator+, 344, 346, 347
operator+=, 347, 348
operator==, 349
save_ini, 349
nntrainer::InputRealizer, 354
~InputRealizer, 355
InputRealizer, 355
realize, 356
nntrainer::internal::PluggedLayer, 496
~PluggedLayer, 498
calcDerivative, 498
calcGradient, 498
exportTo, 498
finalize, 499
forwarding, 499
getType, 499
operator=, 499
PluggedLayer, 497, 498
requireLabel, 500
setBatch, 500
setProperty, 500
supportBackwarding, 500
supportInPlace, 500
nntrainer::internal::PluggedOptimizer, 501
~PluggedOptimizer, 502
applyGradient, 503
getDefaultLearningRate, 503
getOptimizerVariableDim, 503
getType, 504
operator=, 504
PluggedOptimizer, 502
read, 504
save, 505
setProperty, 505
nntrainer::Iteration, 361
batch, 362
begin, 362, 363
end, 363, 364
getInputsRef, 364
getLabelsRef, 365
Iteration, 361
setEndSample, 365
nntrainer::IterationQueue, 368
~IterationQueue, 369
batch, 370
IterationQueue, 369
notifyEndOfRequestEmpty, 370
requestEmptySlot, 370
requestFilledSlot, 370
slots, 371
nntrainer::LayerNode, 377
~LayerNode, 382
calcDerivative, 382
calcGradient, 382
clearOptVar, 382
cloneConfiguration, 382
configureRunContext, 383
executeInPlace, 383
exportTo, 384
finalize, 384
forwarding, 385
getActivationToBeRealized, 386
getActivationType, 386
getDistribute, 387
getExecutionOrder, 387
getFlatten, 388
getInput, 388
getInputConnectionIndex, 388
getInputConnectionName, 389
getInputConnections, 389
getInputDimensions, 389
getInputGrad, 390
getLoss, 390
getName, 390
getNumInputConnections, 391
getNumInputs, 392
getNumOutputConnections, 392
getNumOutputs, 393
getNumWeights, 393
getOutput, 393
getOutputConnection, 394
getOutputConnections, 394
getOutputDimensions, 395
getOutputGrad, 395

getOutputGradUnsafe, 396
getRunContext, 396, 397
getSharedFrom, 397
getTrainable, 397
getType, 398
getWeight, 398
getWeightGrad, 399
getWeightName, 399
getWeightObject, 399
getWeights, 400
getWeightWrapper, 401
hasInputShapeProperty, 401
isFinalized, 401
LayerNode, 381
needsCalcDerivative, 402
needsCalcGradient, 402, 403
PRINT_ALL, 381
PRINT_NONE, 381
PRINT_SUMMARY, 381
PRINT_SUMMARY_META, 381
PrintPreset, 381
printPreset, 403
read, 403
remapConnections, 404
remapIdentifiers, 404
requireLabel, 404
save, 405
setBatch, 405
setExecutionOrder, 406
setInputConnectionIndex, 406
setInputConnectionName, 406
setName, 407
setOutputConnection, 407
setOutputLayers, 408
setProperty, 408
setWeights, 409
supportBackwarding, 409
supportInPlace, 410
nntrainer::LossRealizer, 420
~LossRealizer, 421
LossRealizer, 421
realize, 422
nntrainer::MemoryData< T >, 427
MemoryData, 428, 429
nntrainer::MemoryPlanner, 429
~MemoryPlanner, 430
getType, 430
planLayout, 430
nntrainer::MemoryPool, 431
~MemoryPool, 433
allocate, 433
clear, 433
deallocate, 434
getMemory, 434
isAllocated, 435
MemoryPool, 432
minMemoryRequirement, 435
planLayout, 435
requestMemory, 436
size, 437
nntrainer::MemoryRequest, 438
end, 439
loc, 439
MemoryRequest, 438, 439
offset, 439
size, 440
start, 440
nntrainer::MultioutRealizer, 447
~MultioutRealizer, 448
realize, 448
nntrainer::OptimizedV1Planner, 464
getType, 465
OptimizedV1Planner, 465
planLayout, 465
nntrainer::OptimizedV2Planner, 466
getType, 468
OptimizedV2Planner, 468
planLayout, 468
nntrainer::OptimizedV3Planner, 469
getType, 470
OptimizedV3Planner, 470
planLayout, 470
nntrainer::ParallelBatch, 486
~ParallelBatch, 487
getNumWorkers, 487
ParallelBatch, 486, 487
run, 487
setCallback, 488
nntrainer::PermuteLayer, 492
calcDerivative, 493
exportTo, 494
finalize, 494
forwarding, 494
getType, 495
operator=, 495
PermuteLayer, 493
setProperty, 495
supportBackwarding, 496
nntrainer::PreprocessL2NormLayer, 512
~PreprocessL2NormLayer, 513
calcDerivative, 513
exportTo, 514
finalize, 514
forwarding, 514
getType, 515
PreprocessL2NormLayer, 513
setProperty, 515
supportBackwarding, 515
nntrainer::PreviousInputRealizer, 516
~PreviousInputRealizer, 517
PreviousInputRealizer, 517
realize, 517
nntrainer::profile::GenericProfileListener, 282
~GenericProfileListener, 284
GenericProfileListener, 283
notify, 284

report, 284
reset, 284
result, 285
nntrainer::profile::ProfileEventData, 518
ProfileEventData, 518
nntrainer::profile::ProfileListener, 519
~ProfileListener, 520
notify, 520
ProfileListener, 520
report, 520
reset, 521
result, 521
nntrainer::profile::Profiler, 522
alloc, 523
annotate, 523
dealloc, 524
end, 524
Global, 525
Profiler, 522, 523
registerTimeItem, 525
start, 525
subscribe, 526
unsubscribe, 526
nntrainer::props::Activation, 80
nntrainer::props::ActivationTypeInfo, 83
EnumList, 83
EnumStr, 83
nntrainer::props::AsSequence, 96
nntrainer::props::AverageAttentionWeight, 97
key, 98
prop_tag, 98
nntrainer::props::Axis, 99
isValid, 100
key, 101
prop_tag, 100
nntrainer::props::BasicRegularizer, 106
isValid, 107
nntrainer::props::BasicRegularizerConstant, 107
BasicRegularizerConstant, 109
isValid, 109
key, 110
prop_tag, 109
nntrainer::props::BiasDecay, 111
BiasDecay, 112
key, 113
nntrainer::props::BiasInitializer, 113
nntrainer::props::Bidirectional, 114
Bidirectional, 115
nntrainer::props::BNPARAMS_BETA_INIT, 117
nntrainer::props::BNPARAMS_GAMMA_INIT, 118
nntrainer::props::BNPARAMS_MU_INIT, 120
nntrainer::props::BNPARAMS_VAR_INIT, 121
nntrainer::props::ClipGradByGlobalNorm, 186
key, 188
prop_tag, 188
nntrainer::props::ConcatDimension, 188
nntrainer::props::connection_prop_tag, 197
nntrainer::props::DecayRate, 204
key, 205
prop_tag, 205
nntrainer::props::DecaySteps, 206
key, 207
prop_tag, 207
nntrainer::props::Dilation, 214
Dilation, 216
key, 216
prop_tag, 215
nntrainer::props::DirPath, 221
DirPath, 222
isValid, 223
key, 224
prop_tag, 222
set, 223
nntrainer::props::DisableBias, 224
DisableBias, 226
nntrainer::props::Distribute, 226
nntrainer::props::DropOutRate, 228
DropOutRate, 229
isValid, 229
key, 230
prop_tag, 229
nntrainer::props::DynamicTimeSequence, 230
DynamicTimeSequence, 231
nntrainer::props::Epsilon, 232
Epsilon, 234
isValid, 234
key, 234
prop_tag, 233
nntrainer::props::FilePath, 243
file_size, 245
FilePath, 244
isValid, 245
key, 246
prop_tag, 244
set, 245
nntrainer::props::FilterSize, 247
key, 248
prop_tag, 248
nntrainer::props::Flatten, 270
key, 271
prop_tag, 271
nntrainer::props::FlipDirection, 274
nntrainer::props::FlipDirectionInfo, 275
EnumList, 275
EnumStr, 275
nntrainer::props::GenericShape, 285
key, 287
prop_tag, 287
set, 287
nntrainer::props::HiddenStateActivation, 311
HiddenStateActivation, 312
nntrainer::props::InDim, 313
key, 314
prop_tag, 314
nntrainer::props::InitializerInfo, 329
EnumList, 330

EnumStr, 330
nntrainer::props::InputConnection, 350
InputConnection, 352
key, 352
prop_tag, 351
nntrainer::props::InputIsSequence, 353
nntrainer::props::InputShape, 356
key, 358
prop_tag, 358
nntrainer::props::IntegrateBias, 359
IntegrateBias, 360
nntrainer::props::Iteration, 366
key, 367
prop_tag, 367
nntrainer::props::KernelSize, 372
key, 373
prop_tag, 373
nntrainer::props::LabelLayer, 374
LabelLayer, 375
nntrainer::props::LearningRate, 411
key, 412
prop_tag, 412
nntrainer::props::Loss, 417
isValid, 419
key, 420
Loss, 418
prop_tag, 418
nntrainer::props::MaxTimestep, 426
key, 427
prop_tag, 427
nntrainer::props::MoL_K, 442
key, 443
prop_tag, 443
nntrainer::props::Momentum, 444
isValid, 445
key, 446
Momentum, 445
prop_tag, 445
nntrainer::props::Name, 449
isValid, 451
key, 452
Name, 450
prop_tag, 450
set, 451
nntrainer::props::Normalization, 454
Normalization, 455
nntrainer::props::NumClass, 457
isValid, 459
key, 459
prop_tag, 458
nntrainer::props::NumHeads, 459
key, 461
NumHeads, 461
prop_tag, 460
nntrainer::props::OutDim, 471
key, 473
prop_tag, 472
nntrainer::props::OutputLayer, 473
OutputLayer, 474, 475
nntrainer::props::OutputShape, 475
key, 476
prop_tag, 476
nntrainer::props::Padding1D, 478
compute, 479
isValid, 480
key, 480
prop_tag, 479
nntrainer::props::Padding2D, 481
compute, 482
isValid, 483
key, 483
prop_tag, 482
nntrainer::props::Padding_, 484
prop_tag, 484
nntrainer::props::PermuteDims, 489
isValid, 491
key, 491
prop_tag, 490
nntrainer::props::PoolingType, 506
nntrainer::props::PoolingTypeInfo, 507
EnumList, 508
EnumStr, 508
nntrainer::props::PoolSize, 509
key, 511
PoolSize, 510
prop_tag, 510
nntrainer::props::ProjectedKeyDim, 527
key, 528
prop_tag, 528
nntrainer::props::ProjectedValueDim, 529
key, 530
prop_tag, 530
nntrainer::props::PropsUserData, 544
nntrainer::props::RandomTranslate, 551
key, 553
prop_tag, 552
set, 552
nntrainer::props::RecurrentActivation, 559
RecurrentActivation, 560
nntrainer::props::RecurrentInput, 560
RecurrentInput, 561
nntrainer::props::RecurrentOutput, 562
RecurrentOutput, 563
nntrainer::props::ReduceDimension, 568
nntrainer::props::RegularizerInfo, 569
EnumList, 570
nntrainer::props::ResetAfter, 573
key, 575
prop_tag, 575
ResetAfter, 575
nntrainer::props::ReturnAttentionWeight, 577
key, 579
prop_tag, 578
ReturnAttentionWeight, 579
nntrainer::props::ReturnAttentionWeightInfo, 579
EnumList, 580

nntrainer::props::ReturnSequences, 581
ReturnSequences, 582
nntrainer::props::SharedFrom, 622
key, 623
prop_tag, 623
nntrainer::props::SplitDimension, 633
isValid, 634
nntrainer::props::SplitNumber, 635
key, 636
prop_tag, 636
nntrainer::props::Standardization, 640
Standardization, 641
nntrainer::props::Stride, 642
key, 643
prop_tag, 643
Stride, 643
nntrainer::props::TargetShape, 650
key, 652
prop_tag, 651
nntrainer::props::Timestep, 685
key, 686
prop_tag, 686
nntrainer::props::Trainable, 687
Trainable, 688
nntrainer::props::Unit, 692
key, 693
prop_tag, 693
nntrainer::props::UnrollFor, 695
nntrainer::props::WeightDecay, 725
key, 726
WeightDecay, 726
nntrainer::props::WeightInitializer, 727
nntrainer::props::WeightRegularizer, 728
nntrainer::props::WeightRegularizerConstant, 730
key, 731
WeightRegularizerConstant, 731
nntrainer::props::ZeroIdxMask, 736
key, 737
prop_tag, 737
nntrainer::PropsBufferSize, 531
key, 532
prop_tag, 532
PropsBufferSize, 532
nntrainer::PropsMax, 533
key, 534
prop_tag, 534
PropsMax, 534
nntrainer::PropsMin, 535
key, 536
prop_tag, 536
PropsMin, 536
nntrainer::PropsNNSModelPath, 537
isValid, 538
key, 538
prop_tag, 538
nntrainer::PropsNumSamples, 539
key, 541
prop_tag, 540
PropsNumSamples, 540
nntrainer::PropsTflModelPath, 541
isValid, 543
key, 543
prop_tag, 542
nntrainer::RandomDataOneHotProducer, 547
~RandomDataOneHotProducer, 548
finalize, 549
getType, 549
isMultiThreadSafe, 549
RandomDataOneHotProducer, 548
setProperty, 550
size, 550
nntrainer::RawFileDataProducer, 554
~RawFileDataProducer, 556
exportTo, 556
finalize, 556
getType, 557
pixel_size, 558
RawFileDataProducer, 555, 556
setProperty, 557
size, 557
nntrainer::RecurrentRealizer, 564
~RecurrentRealizer, 566
realize, 566
RecurrentRealizer, 565, 566
nntrainer::RemapRealizer, 571
~RemapRealizer, 572
realize, 573
RemapRealizer, 572
nntrainer::RunLayerContext, 584
executeInPlace, 587
getIncomingDerivative, 587
getInput, 588, 589
getInputGrad, 589
getLabel, 590
getLoss, 590
getName, 591
getNumInputs, 591
getNumOutputs, 591
getNumTensors, 592
getNumWeightOptVar, 592
getNumWeights, 593
getOutgoingDerivative, 593
getOutput, 594, 595
getOutputGrad, 595
getOutputGradUnsafe, 596
getRegularizationLoss, 597
getTensor, 598
getTensorGrad, 599
getTensorName, 599
getTrainable, 600
getWeight, 600
getWeightGrad, 601
getWeightName, 601
getWeightObject, 602
getWeightOptVar, 602
inputHasGradient, 603

isGradientClipByGlobalNorm, 603
isGradientFirstAccess, 604
isGradientLastAccess, 604
isLabelAvailable, 605
isWeightDependent, 605
outputHasGradient, 606
readyToUse, 606
RunLayerContext, 586
setBatch, 607
setLoss, 607
tensorHasGradient, 607
updateTensor, 608
validate, 608
weightHasGradient, 609
nntrainer::RunOptimizerContext, 610
applyGradient, 610
getGradient, 611
getIteration, 611
getLearningRate, 612
getOptimizerVariable, 612
getWeight, 613
readyToUse, 613
RunOptimizerContext, 610
nntrainer::Sample, 614
getInputsRef, 615
getLabelsRef, 615
Sample, 614
nntrainer::ScopedView< T >, 616
~ScopedView, 618
get, 618
isEmpty, 618
ScopedView, 617
nntrainer::SliceRealizer, 626
~SliceRealizer, 627
realize, 627
SliceRealizer, 627
nntrainer::SrcSharedTensor, 639
nntrainer::SwapDevice, 646
~SwapDevice, 647
finish, 647
getBuffer, 648
getDevicePath, 648
isOperating, 648
putBuffer, 649
start, 649
swap_device_default_path, 649
SwapDevice, 647
nntrainer::Task, 652
~Task, 653
done, 654
getData, 654
getType, 654
getWork, 654
setState, 655
started, 655
Task, 653
nntrainer::TaskAsync< T >, 656
~TaskAsync, 657
getPriority, 657
getTimeout, 658
getType, 658
setPriority, 658
setTimeout, 659
TaskAsync, 657
nntrainer::TaskExecutor, 659
~TaskExecutor, 661
cancel, 662
clean, 662
CompleteCallback, 661
handleWork, 662
run, 663, 664
TaskExecutor, 661
TaskInfo, 661
worker, 664
nntrainer::Tensor::BroadcastInfo, 125
BroadcastInfo, 125
buffer_axis, 125
buffer_size, 125
strides, 126
nntrainer::TensorSpecV2, 666
additional_exec_order, 668
dim, 668
initializer, 668
ls, 668
MAYBE_MODIFYING_VIEW, 667
name, 668
offset, 669
PLACEHOLDER, 667
READ_ONLY_VIEW, 667
reference_name, 669
request_type, 669
RequestType, 667
SHARED, 667
UNIQUE, 667
nntrainer::TfliteInterpreter, 670
~TfliteInterpreter, 671
deserialize, 671
serialize, 672
TfliteInterpreter, 671
nntrainer::TfOpNode, 673
arg, 674
arity, 675
finalize, 675
getBuiltinOps, 676
getInputNodes, 676
getInputs, 676
getOptionType, 677
getOpType, 677
getOutputs, 677
getWeights, 678
isInputNode, 678
isNeedReorder, 679
isOutputNode, 679
isTrainable, 679
isVirtualNode, 679
setArg, 680

setBuiltinOptions, 680
setInputTransformFn, 680
setLayerNode, 681
setNeedReorderWeight, 682
setOpType, 682
setTrainable, 683
setWeightTransformFn, 683
TfOpNode, 674
weightReorder, 683
nntrainer::Var_Grad, 696
getDim, 701
getGradient, 701
getGradientName, 701
getGradientNorm, 701
getGradientRef, 702
getName, 702
getVariable, 703
getVariableRef, 703, 704
grad, 708
hasGradient, 704
initializeGradient, 705
initializeVariable, 705
is_dependent, 708
is_first_access_gradient, 708
is_last_access_gradient, 709
isDependent, 706
isGradientFirstAccess, 706
isGradientLastAccess, 706
operator=, 706, 707
resetGradient, 707
setAsGradientFirstAccess, 707
setAsGradientLastAccess, 707
setBatchSize, 708
Spec, 698
var, 709
Var_Grad, 698–700
nntrainer::VarGradSpecV2, 709
gradient_spec, 711
operator=, 711
VarGradSpecV2, 710, 711
variable_spec, 711
nntrainer::ViewQueue< T >, 712
isEmpty, 713
push, 713
size, 713
waitAndPop, 714
nntrainer::Weight, 715
clipGradientByGlobalNorm, 720
clone, 720
getNumOptVariable, 720
getOptimizerVariableRef, 720
isGradientClipByGlobalNorm, 721
isWeightDecay, 722
isWeightRegularizerL2Norm, 722
operator=, 723
setOptimizerVariables, 724
Spec, 717
swap, 724
Weight, 717–719
nntrainer::WeightSpecV2, 732
clip_by_global_norm, 732
decay, 733
regularizer, 733
regularizer_constant, 733
vg_spec, 733
nntrainer::WGradMemoryRequest, 734
nntrainer_error.h, 906
ESTRPIPE, 908
ml_error_e, 908
ML_ERROR_INVALID_PARAMETER, 909
ML_ERROR_NONE, 909
ML_ERROR_NOT_SUPPORTED, 909
ML_ERROR_OUT_OF_MEMORY, 909
ML_ERROR_PERMISSION_DENIED, 909
ML_ERROR_TIMED_OUT, 909
ML_ERROR_TRY_AGAIN, 909
ML_ERROR_UNKNOWN, 909
NNTR_THROW_IF, 908
NNTR_THROW_IF_CLEANUP, 908
nntrainer_log.h, 909
ml_logd, 910
ml_loge, 910
ml_logi, 911
ml_logw, 911
nntrainer_logger.cpp, 912
nntrainer_logger.h, 913
NO_READ_CONSIST
nntrainer, 60
NO_WRITE_BACK
nntrainer, 60
NON_RESTRICTING
nntrainer, 62
NONE
nntrainer, 62, 64
nntrainer::CacheElem, 162
Normalization
nntrainer::props::Normalization, 455
notify
nntrainer::profile::GenericProfileListener, 284
nntrainer::profile::ProfileListener, 520
notifyEndOfRequestEmpty
nntrainer::IterationQueue, 370
npu
nntrainer::Device, 212
NumHeads
nntrainer::props::NumHeads, 461

offset
nntrainer::MemoryRequest, 439
nntrainer::TensorSpecV2, 669
Op, 462
operator!=
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 302
nntrainer::IniSection, 323
nntrainer::IniWrapper, 344
operator<<

nntrainer::exception::ErrorNotification< Err, >, 237
nntrainer::IniSection, 328
nntrainer::IniWrapper, 350
operator*
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 298
nntrainer::GraphNodeReverseIterator< T_iterator >, 305
operator()
std::hash< nntrainer::Connection >, 310
operator+
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 298
nntrainer::IniSection, 324, 325
nntrainer::IniWrapper, 344, 346, 347
operator++
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 299
operator+=
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 299
nntrainer::IniSection, 325, 326
nntrainer::IniWrapper, 347, 348
operator-
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 300
operator->
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 302
nntrainer::GraphNodeReverseIterator< T_iterator >, 306
operator--
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 301
operator-=
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 301
operator=
nntrainer::Connection, 194, 195
nntrainer::internal::PluggedLayer, 499
nntrainer::internal::PluggedOptimizer, 504
nntrainer::PermuteLayer, 495
nntrainer::Var_Grad, 706, 707
nntrainer::VarGradSpecV2, 711
nntrainer::Weight, 723
operator==
nntrainer::Connection, 195
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 302
nntrainer::IniSection, 327
nntrainer::IniWrapper, 349
OptimizedV1Planner
nntrainer::OptimizedV1Planner, 465
OptimizedV2Planner
nntrainer::OptimizedV2Planner, 468
OptimizedV3Planner
nntrainer::OptimizedV3Planner, 470
optimizers/adam.cpp, 914
optimizers/adam.h, 915
optimizers/lr_scheduler.h, 915
optimizers/lr_scheduler_constant.cpp, 916
optimizers/lr_scheduler_constant.h, 917
optimizers/lr_scheduler_exponential.cpp, 918
optimizers/lr_scheduler_exponential.h, 919
optimizers/lr_scheduler_step.cpp, 919
optimizers/lr_scheduler_step.h, 921
optimizers/optimizer_context.h, 922
optimizers/optimizer_devel.cpp, 923
optimizers/optimizer_devel.h, 924
optimizers/optimizer_wrapped.cpp, 925
optimizers/optimizer_wrapped.h, 926
optimizers/plugged_optimizer.h, 927
optimizers/sgd.cpp, 929
optimizers/sgd.h, 930
Options
nntrainer::CacheElem, 162
OUTPUT
nntrainer, 62
outputHasGradient
nntrainer::RunLayerContext, 606
OutputLayer
nntrainer::props::OutputLayer, 474, 475
outSpec
nntrainer::InitLayerContext, 335
overlap
nntrainer, 72

ParallelBatch
nntrainer::ParallelBatch, 486, 487
parse_properties
nntrainer, 72
PermuteLayer
nntrainer::PermuteLayer, 493
pixel_size
nntrainer::RawFileDataProducer, 558
PLACEHOLDER
nntrainer::TensorSpecV2, 667
planLayout
nntrainer::BasicPlanner, 105
nntrainer::MemoryPlanner, 430
nntrainer::MemoryPool, 435
nntrainer::OptimizedV1Planner, 465
nntrainer::OptimizedV2Planner, 468
nntrainer::OptimizedV3Planner, 470
PluggedLayer
nntrainer::internal::PluggedLayer, 497, 498
PluggedOptimizer
nntrainer::internal::PluggedOptimizer, 502
PoolSize
nntrainer::props::PoolSize, 510
PreprocessL2NormLayer
nntrainer::PreprocessL2NormLayer, 513
PreviousInputRealizer
nntrainer::PreviousInputRealizer, 517
print
nntrainer::IniSection, 327
PRINT_ALL

nntrainer::LayerNode, 381
PRINT_INST_INFO
nntrainer, 63
PRINT_METRIC
nntrainer, 63
PRINT_NONE
nntrainer::LayerNode, 381
PRINT_PROP
nntrainer, 63
PRINT_PROP_META
nntrainer, 63
PRINT_SHAPE_INFO
nntrainer, 63
PRINT_SUMMARY
nntrainer::LayerNode, 381
PRINT_SUMMARY_META
nntrainer::LayerNode, 381
PRINT_WEIGHTS
nntrainer, 63
PrintOption
nntrainer, 63
PrintPreset
nntrainer::LayerNode, 381
printPreset
nntrainer::LayerNode, 403
ProfileEventData
nntrainer::profile::ProfileEventData, 518
ProfileListener
nntrainer::profile::ProfileListener, 520
Profiler
nntrainer::profile::Profiler, 522, 523
prop_tag
nntrainer::props::AverageAttentionWeight, 98
nntrainer::props::Axis, 100
nntrainer::props::BasicRegularizerConstant, 109
nntrainer::props::ClipGradByGlobalNorm, 188
nntrainer::props::DecayRate, 205
nntrainer::props::DecaySteps, 207
nntrainer::props::Dilation, 215
nntrainer::props::DirPath, 222
nntrainer::props::DropOutRate, 229
nntrainer::props::Epsilon, 233
nntrainer::props::FilePath, 244
nntrainer::props::FilterSize, 248
nntrainer::props::Flatten, 271
nntrainer::props::GenericShape, 287
nntrainer::props::InDim, 314
nntrainer::props::InputConnection, 351
nntrainer::props::InputShape, 358
nntrainer::props::Iteration, 367
nntrainer::props::KernelSize, 373
nntrainer::props::LearningRate, 412
nntrainer::props::Loss, 418
nntrainer::props::MaxTimestep, 427
nntrainer::props::MoL_K, 443
nntrainer::props::Momentum, 445
nntrainer::props::Name, 450
nntrainer::props::NumClass, 458
nntrainer::props::NumHeads, 460
nntrainer::props::OutDim, 472
nntrainer::props::OutputShape, 476
nntrainer::props::Padding1D, 479
nntrainer::props::Padding2D, 482
nntrainer::props::Padding_, 484
nntrainer::props::PermuteDims, 490
nntrainer::props::PoolSize, 510
nntrainer::props::ProjectedKeyDim, 528
nntrainer::props::ProjectedValueDim, 530
nntrainer::props::RandomTranslate, 552
nntrainer::props::ResetAfter, 575
nntrainer::props::ReturnAttentionWeight, 578
nntrainer::props::SharedFrom, 623
nntrainer::props::SplitNumber, 636
nntrainer::props::Stride, 643
nntrainer::props::TargetShape, 651
nntrainer::props::Timestep, 686
nntrainer::props::Unit, 693
nntrainer::props::ZeroIdxMask, 737
nntrainer::PropsBufferSize, 532
nntrainer::PropsMax, 534
nntrainer::PropsMin, 536
nntrainer::PropsNNSModelPath, 538
nntrainer::PropsNumSamples, 540
nntrainer::PropsTflModelPath, 542
propagateTimestep
nntrainer, 73
PropsBufferSize
nntrainer::PropsBufferSize, 532
PropsMax
nntrainer::PropsMax, 534
PropsMin
nntrainer::PropsMin, 536
PropsNumSamples
nntrainer::PropsNumSamples, 540
push
nntrainer::ViewQueue< T >, 713
putBuffer
nntrainer::SwapDevice, 649

QUERY
nntrainer, 62

RandomDataOneHotProducer
nntrainer::RandomDataOneHotProducer, 548
RawFileDataProducer
nntrainer::RawFileDataProducer, 555, 556
read
nntrainer::internal::PluggedOptimizer, 504
nntrainer::LayerNode, 403
READ_CONSIST
nntrainer, 60
READ_ONLY_VIEW
nntrainer::TensorSpecV2, 667
readyToUse
nntrainer::RunLayerContext, 606
nntrainer::RunOptimizerContext, 613
realize

nntrainer::ActivationRealizer, 82
nntrainer::BnRealizer, 124
nntrainer::FlattenRealizer, 273
nntrainer::GraphRealizer, 308
nntrainer::InputRealizer, 356
nntrainer::LossRealizer, 422
nntrainer::MultioutRealizer, 448
nntrainer::PreviousInputRealizer, 517
nntrainer::RecurrentRealizer, 566
nntrainer::RemapRealizer, 573
nntrainer::SliceRealizer, 627
RecurrentActivation
nntrainer::props::RecurrentActivation, 560
RecurrentInput
nntrainer::props::RecurrentInput, 561
RecurrentOutput
nntrainer::props::RecurrentOutput, 563
RecurrentRealizer
nntrainer::RecurrentRealizer, 565, 566
reference_name
nntrainer::TensorSpecV2, 669
registerFactory
nntrainer::AppContext, 89, 90
registerLayer
nntrainer::AppContext, 91
registerOptimizer
nntrainer::AppContext, 91
registerPluggableFromDirectory
nntrainer::AppContext, 92
registerTimeItem
nntrainer::profile::Profiler, 525
regularizer
nntrainer::WeightSpecV2, 733
regularizer_constant
nntrainer::WeightSpecV2, 733
remapConnections
nntrainer::LayerNode, 404
remapIdentifiers
nntrainer::LayerNode, 404
RemapRealizer
nntrainer::RemapRealizer, 572
report
nntrainer::profile::GenericProfileListener, 284
nntrainer::profile::ProfileListener, 520
request_type
nntrainer::TensorSpecV2, 669
requestEmptySlot
nntrainer::IterationQueue, 370
requestFilledSlot
nntrainer::IterationQueue, 370
requestMemory
nntrainer::CachePool, 178
nntrainer::MemoryPool, 436
requestOutputs
nntrainer::InitLayerContext, 336
requestTensor
nntrainer::InitLayerContext, 336, 337
RequestType
nntrainer::TensorSpecV2, 667
requestWeight
nntrainer::InitLayerContext, 337, 338
requireLabel
nntrainer::CentroidKNN, 185
nntrainer::internal::PluggedLayer, 500
nntrainer::LayerNode, 404
reset
nntrainer::CacheElem, 163
nntrainer::profile::GenericProfileListener, 284
nntrainer::profile::ProfileListener, 521
ResetAfter
nntrainer::props::ResetAfter, 575
resetGradient
nntrainer::Var_Grad, 707
RESTRICTING
nntrainer, 62
result
nntrainer::profile::GenericProfileListener, 285
nntrainer::profile::ProfileListener, 521
ReturnAttentionWeight
nntrainer::props::ReturnAttentionWeight, 579
ReturnSequences
nntrainer::props::ReturnSequences, 582
rng
nntrainer, 77
run
nntrainer::ParallelBatch, 487
nntrainer::TaskExecutor, 663, 664
RunLayerContext
nntrainer::RunLayerContext, 586
RunOptimizerContext
nntrainer::RunOptimizerContext, 610

Sample
nntrainer::Sample, 614
save
nntrainer::internal::PluggedOptimizer, 505
nntrainer::LayerNode, 405
save_ini
nntrainer::IniWrapper, 349
saveResult
nntrainer::Exporter, 240
ScopedView
nntrainer::ScopedView< T >, 617
serialize
nntrainer::FlatBufferInterpreter, 250
nntrainer::GraphInterpreter, 291
nntrainer::IniGraphInterpreter, 317
nntrainer::TfliteInterpreter, 672
serialize_v1
nntrainer::GraphInterpreter, 291
set
nntrainer::props::DirPath, 223
nntrainer::props::FilePath, 245
nntrainer::props::GenericShape, 287
nntrainer::props::Name, 451
nntrainer::props::RandomTranslate, 552
setArg

nntrainer::FlatBufferOpNode, 257
nntrainer::TfOpNode, 680
setAsGradientFirstAccess
nntrainer::Var_Grad, 707
setAsGradientLastAccess
nntrainer::Var_Grad, 707
setBatch
nntrainer::internal::PluggedLayer, 500
nntrainer::LayerNode, 405
nntrainer::RunLayerContext, 607
setBatchSize
nntrainer::Var_Grad, 708
setBuiltinOptions
nntrainer::FlatBufferOpNode, 257
nntrainer::TfOpNode, 680
setCallback
nntrainer::ParallelBatch, 488
setDevice
nntrainer::Device, 213
setDynDimFlagInputDimension
nntrainer::InitLayerContext, 339
setEffDimFlagInputDimension
nntrainer::InitLayerContext, 339
setEndSample
nntrainer::Iteration, 365
setEntry
nntrainer::IniSection, 328
setExecutionOrder
nntrainer::GraphNode, 295
nntrainer::LayerNode, 406
setInplaceSharedMemoryConfigByLayer
nntrainer, 74
setInputConnectionIndex
nntrainer::LayerNode, 406
setInputConnectionName
nntrainer::LayerNode, 406
setInputTransformFn
nntrainer::TfOpNode, 680
setLayerNode
nntrainer::FlatBufferOpNode, 257
nntrainer::TfOpNode, 681
setLoss
nntrainer::RunLayerContext, 607
setMemoryFraction
nntrainer::Device, 214
setName
nntrainer::GraphNode, 295
nntrainer::LayerNode, 407
setNeedReorderWeight
nntrainer::TfOpNode, 682
setNumThreads
nntrainer::Backend, 103
setOptimizerVariables
nntrainer::Weight, 724
setOpType
nntrainer::FlatBufferOpNode, 258
nntrainer::TfOpNode, 682
setOutputConnection
nntrainer::LayerNode, 407
setOutputDimensions
nntrainer::InitLayerContext, 339
setOutputLayers
nntrainer::LayerNode, 408
setPriority
nntrainer::TaskAsync< T >, 658
setProperty
nntrainer::CentroidKNN, 185
nntrainer::DataProducer, 203
nntrainer::DirDataProducer, 220
nntrainer::FuncDataProducer, 280
nntrainer::internal::PluggedLayer, 500
nntrainer::internal::PluggedOptimizer, 505
nntrainer::LayerNode, 408
nntrainer::PermuteLayer, 495
nntrainer::PreprocessL2NormLayer, 515
nntrainer::RandomDataOneHotProducer, 550
nntrainer::RawFileDataProducer, 557
setState
nntrainer::Task, 655
setTimeout
nntrainer::TaskAsync< T >, 659
setTrainable
nntrainer::TfOpNode, 683
setWeights
nntrainer::LayerNode, 409
setWeightTransformFn
nntrainer::TfOpNode, 683
setWorkingDirectory
nntrainer::AppContext, 92
sgemv_loop
blas_interface.cpp, 935
SHARED
nntrainer::TensorSpecV2, 667
size
nntrainer::DataProducer, 203
nntrainer::DirDataProducer, 220
nntrainer::MemoryPool, 437
nntrainer::MemoryRequest, 440
nntrainer::RandomDataOneHotProducer, 550
nntrainer::RawFileDataProducer, 557
nntrainer::ViewQueue< T >, 713
SIZE_UNDEFINED
nntrainer::DataProducer, 204
SliceRealizer
nntrainer::SliceRealizer, 627
slots
nntrainer::IterationQueue, 371
Spec
nntrainer::Var_Grad, 698
nntrainer::Weight, 717
Standardization
nntrainer::props::Standardization, 641
start
nntrainer::MemoryRequest, 440
nntrainer::profile::Profiler, 525
nntrainer::SwapDevice, 649

started
nntrainer::Task, 655
std::hash< nntrainer::Connection >, 310
operator(), 310
Stride
nntrainer::props::Stride, 643
strides
nntrainer::Tensor::BroadcastInfo, 126
subscribe
nntrainer::profile::Profiler, 526
suffixSpec
nntrainer, 74
supportBackwarding
nntrainer::CentroidKNN, 186
nntrainer::internal::PluggedLayer, 500
nntrainer::LayerNode, 409
nntrainer::PermuteLayer, 496
nntrainer::PreprocessL2NormLayer, 515
supportInPlace
nntrainer::internal::PluggedLayer, 500
nntrainer::LayerNode, 410
swap
nntrainer::Weight, 724
swap_device_default_path
nntrainer::SwapDevice, 649
SwapDevice
nntrainer::SwapDevice, 647
swapIn
nntrainer::CacheElem, 163
swapOut
nntrainer::CacheElem, 164

Task
nntrainer::Task, 653
TaskAsync
nntrainer::TaskAsync< T >, 657
TaskExecutor
nntrainer::TaskExecutor, 661
TaskInfo
nntrainer::TaskExecutor, 661
TEMPORAL
nntrainer, 60
tensor.cpp
CREATE_IF_EMPTY_DIMS, 963
transposeloop, 963
tensor/basic_planner.cpp, 931
tensor/basic_planner.h, 932
tensor/blas_interface.cpp, 933
tensor/blas_interface.h, 935
tensor/cache_elem.cpp, 936
tensor/cache_elem.h, 937
tensor/cache_loader.cpp, 938
tensor/cache_loader.h, 939
tensor/cache_pool.cpp, 941
tensor/cache_pool.h, 942
tensor/lazy_tensor.cpp, 943
tensor/lazy_tensor.h, 944
tensor/manager.cpp, 945
tensor/manager.h, 946
tensor/memory_data.h, 948
tensor/memory_planner.h, 949
tensor/memory_pool.cpp, 950
tensor/memory_pool.h, 951
tensor/optimized_v1_planner.cpp, 953
tensor/optimized_v2_planner.cpp, 954
tensor/optimized_v3_planner.cpp, 955
tensor/swap_device.cpp, 956
tensor/task.h, 958
tensor/task_executor.cpp, 959
tensor/task_executor.h, 960
tensor/tensor.cpp, 962
tensor/tensor.h, 964
tensor/tensor_dim.cpp, 965
tensor/tensor_pool.cpp, 966
tensor/tensor_pool.h, 967
tensor/tensor_wrap_specs.h, 968
tensor/var_grad.cpp, 969
tensor/var_grad.h, 971
tensor/weight.cpp, 972
tensor/weight.h, 973
tensorHasGradient
nntrainer::RunLayerContext, 607
TensorLifespan
nntrainer, 63
TensorSpec
nntrainer::InitLayerContext, 332
tflite::AbsOptionsBuilder, 79
tflite::AddNOptionsBuilder, 84
tflite::AddOptionsBuilder, 84
tflite::ArgMaxOptionsBuilder, 95
tflite::ArgMinOptionsBuilder, 95
tflite::BatchMatMulOptionsBuilder, 110
tflite::BatchToSpaceNDOptionsBuilder, 111
tflite::BidirectionalSequenceLSTMOptionsBuilder, 116
tflite::BidirectionalSequenceRNNOptionsBuilder, 116
tflite::BufferBuilder, 126
tflite::BuiltinOptionsTraits< T >, 127
tflite::BuiltinOptionsTraits< tflite::AbsOptions >, 127
tflite::BuiltinOptionsTraits< tflite::AddNOptions >, 127
tflite::BuiltinOptionsTraits< tflite::AddOptions >, 128
tflite::BuiltinOptionsTraits< tflite::ArgMaxOptions >, 128
tflite::BuiltinOptionsTraits< tflite::ArgMinOptions >, 128
tflite::BuiltinOptionsTraits< tflite::BatchMatMulOptions >, 129
tflite::BuiltinOptionsTraits< tflite::BatchToSpaceNDOptions >, 129
tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceLSTMOptions >, 129
tflite::BuiltinOptionsTraits< tflite::BidirectionalSequenceRNNOptions >, 130
tflite::BuiltinOptionsTraits< tflite::CallOptions >, 130
tflite::BuiltinOptionsTraits< tflite::CastOptions >, 130
tflite::BuiltinOptionsTraits< tflite::ConcatEmbeddingsOptions >, 131
tflite::BuiltinOptionsTraits< tflite::ConcatenationOptions >, 131
tflite::BuiltinOptionsTraits< tflite::Conv2DOptions >, 131

tflite::BuiltinOptionsTraits< tflite::CosOptions >, 132
tflite::BuiltinOptionsTraits< tflite::CumsumOptions >, 132
tflite::BuiltinOptionsTraits< tflite::DensifyOptions >, 132
tflite::BuiltinOptionsTraits< tflite::DepthToSpaceOptions >, 133
tflite::BuiltinOptionsTraits< tflite::DepthwiseConv2DOptions >, 133
tflite::BuiltinOptionsTraits< tflite::DequantizeOptions >, 133
tflite::BuiltinOptionsTraits< tflite::DivOptions >, 134
tflite::BuiltinOptionsTraits< tflite::EmbeddingLookupSparseOptions >, 134
tflite::BuiltinOptionsTraits< tflite::EqualOptions >, 134
tflite::BuiltinOptionsTraits< tflite::ExpandDimsOptions >, 135
tflite::BuiltinOptionsTraits< tflite::ExpOptions >, 135
tflite::BuiltinOptionsTraits< tflite::FakeQuantOptions >, 135
tflite::BuiltinOptionsTraits< tflite::FillOptions >, 136
tflite::BuiltinOptionsTraits< tflite::FloorDivOptions >, 136
tflite::BuiltinOptionsTraits< tflite::FloorModOptions >, 136
tflite::BuiltinOptionsTraits< tflite::FullyConnectedOptions >, 137
tflite::BuiltinOptionsTraits< tflite::GatherNdOptions >, 137
tflite::BuiltinOptionsTraits< tflite::GatherOptions >, 137
tflite::BuiltinOptionsTraits< tflite::GreaterEqualOptions >, 138
tflite::BuiltinOptionsTraits< tflite::GreaterOptions >, 138
tflite::BuiltinOptionsTraits< tflite::HardSwishOptions >, 138
tflite::BuiltinOptionsTraits< tflite::IfOptions >, 139
tflite::BuiltinOptionsTraits< tflite::L2NormOptions >, 139
tflite::BuiltinOptionsTraits< tflite::LeakyReluOptions >, 139
tflite::BuiltinOptionsTraits< tflite::LessEqualOptions >, 140
tflite::BuiltinOptionsTraits< tflite::LessOptions >, 140
tflite::BuiltinOptionsTraits< tflite::LocalResponseNormalizationOptions >, 140
tflite::BuiltinOptionsTraits< tflite::LogicalAndOptions >, 141
tflite::BuiltinOptionsTraits< tflite::LogicalNotOptions >, 141
tflite::BuiltinOptionsTraits< tflite::LogicalOrOptions >, 141
tflite::BuiltinOptionsTraits< tflite::LogSoftmaxOptions >, 142
tflite::BuiltinOptionsTraits< tflite::LSHProjectionOptions >, 142
tflite::BuiltinOptionsTraits< tflite::LSTMOptions >, 142
tflite::BuiltinOptionsTraits< tflite::MatrixDiagOptions >, 143
tflite::BuiltinOptionsTraits< tflite::MatrixSetDiagOptions >, 143
tflite::BuiltinOptionsTraits< tflite::MaximumMinimumOptions >, 143
tflite::BuiltinOptionsTraits< tflite::MirrorPadOptions >, 144
tflite::BuiltinOptionsTraits< tflite::MulOptions >, 144
tflite::BuiltinOptionsTraits< tflite::NegOptions >, 144
tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV4Options >, 145
tflite::BuiltinOptionsTraits< tflite::NonMaxSuppressionV5Options >, 145
tflite::BuiltinOptionsTraits< tflite::NotEqualOptions >, 145
tflite::BuiltinOptionsTraits< tflite::OneHotOptions >, 146
tflite::BuiltinOptionsTraits< tflite::PackOptions >, 146
tflite::BuiltinOptionsTraits< tflite::PadOptions >, 146
tflite::BuiltinOptionsTraits< tflite::PadV2Options >, 147
tflite::BuiltinOptionsTraits< tflite::Pool2DOptions >, 147
tflite::BuiltinOptionsTraits< tflite::PowOptions >, 147
tflite::BuiltinOptionsTraits< tflite::QuantizeOptions >, 148
tflite::BuiltinOptionsTraits< tflite::RangeOptions >, 148
tflite::BuiltinOptionsTraits< tflite::RankOptions >, 148
tflite::BuiltinOptionsTraits< tflite::ReducerOptions >, 149
tflite::BuiltinOptionsTraits< tflite::ReshapeOptions >, 149
tflite::BuiltinOptionsTraits< tflite::ResizeBilinearOptions >, 149
tflite::BuiltinOptionsTraits< tflite::ResizeNearestNeighborOptions >, 150
tflite::BuiltinOptionsTraits< tflite::ReverseSequenceOptions >, 150
tflite::BuiltinOptionsTraits< tflite::ReverseV2Options >, 150
tflite::BuiltinOptionsTraits< tflite::RNNOptions >, 151
tflite::BuiltinOptionsTraits< tflite::ScatterNdOptions >, 151
tflite::BuiltinOptionsTraits< tflite::SegmentSumOptions >, 151
tflite::BuiltinOptionsTraits< tflite::SelectOptions >, 152
tflite::BuiltinOptionsTraits< tflite::SelectV2Options >, 152
tflite::BuiltinOptionsTraits< tflite::SequenceRNNOptions >, 152
tflite::BuiltinOptionsTraits< tflite::ShapeOptions >, 153
tflite::BuiltinOptionsTraits< tflite::SkipGramOptions >, 153
tflite::BuiltinOptionsTraits< tflite::SliceOptions >, 153
tflite::BuiltinOptionsTraits< tflite::SoftmaxOptions >, 154
tflite::BuiltinOptionsTraits< tflite::SpaceToBatchNDOptions >, 154
tflite::BuiltinOptionsTraits< tflite::SpaceToDepthOptions >, 154
tflite::BuiltinOptionsTraits< tflite::SparseToDenseOptions >, 155
tflite::BuiltinOptionsTraits< tflite::SplitOptions >, 155
tflite::BuiltinOptionsTraits< tflite::SplitVOptions >, 155

tflite::BuiltinOptionsTraits< tflite::SquaredDifferenceOptions >, 156
tflite::BuiltinOptionsTraits< tflite::SquareOptions >, 156
tflite::BuiltinOptionsTraits< tflite::SqueezeOptions >, 156
tflite::BuiltinOptionsTraits< tflite::StridedSliceOptions >, 157
tflite::BuiltinOptionsTraits< tflite::SubOptions >, 157
tflite::BuiltinOptionsTraits< tflite::SVDFOptions >, 157
tflite::BuiltinOptionsTraits< tflite::TileOptions >, 158
tflite::BuiltinOptionsTraits< tflite::TopKV2Options >, 158
tflite::BuiltinOptionsTraits< tflite::TransposeConvOptions >, 158
tflite::BuiltinOptionsTraits< tflite::TransposeOptions >, 159
tflite::BuiltinOptionsTraits< tflite::UnidirectionalSequenceLSTMOptions >, 159
tflite::BuiltinOptionsTraits< tflite::UniqueOptions >, 159
tflite::BuiltinOptionsTraits< tflite::UnpackOptions >, 160
tflite::BuiltinOptionsTraits< tflite::WhereOptions >, 160
tflite::BuiltinOptionsTraits< tflite::WhileOptions >, 160
tflite::BuiltinOptionsTraits< tflite::ZerosLikeOptions >, 161
tflite::CallOptionsBuilder, 180
tflite::CastOptionsBuilder, 181
tflite::ConcatEmbeddingsOptionsBuilder, 190
tflite::ConcatenationOptionsBuilder, 190
tflite::Conv2DOptionsBuilder, 197
tflite::CosOptionsBuilder, 198
tflite::CumsumOptionsBuilder, 198
tflite::CustomQuantizationBuilder, 199
tflite::DensifyOptionsBuilder, 208
tflite::DepthToSpaceOptionsBuilder, 209
tflite::DepthwiseConv2DOptionsBuilder, 210
tflite::DequantizeOptionsBuilder, 210
tflite::DimensionMetadataBuilder, 216
tflite::DivOptionsBuilder, 227
tflite::EmbeddingLookupSparseOptionsBuilder, 232
tflite::EqualOptionsBuilder, 235
tflite::ExpandDimsOptionsBuilder, 237
tflite::ExpOptionsBuilder, 238
tflite::FakeQuantOptionsBuilder, 242
tflite::FillOptionsBuilder, 246
tflite::FLATBUFFERS_FINAL_CLASS, 259
tflite::FloorDivOptionsBuilder, 276
tflite::FloorModOptionsBuilder, 276
tflite::FullyConnectedOptionsBuilder, 277
tflite::GatherNdOptionsBuilder, 281
tflite::GatherOptionsBuilder, 282
tflite::GreaterEqualOptionsBuilder, 308
tflite::GreaterOptionsBuilder, 309
tflite::HardSwishOptionsBuilder, 309
tflite::IfOptionsBuilder, 313
tflite::Int32VectorBuilder, 358
tflite::L2NormOptionsBuilder, 373
tflite::LeakyReluOptionsBuilder, 411
tflite::LessEqualOptionsBuilder, 413
tflite::LessOptionsBuilder, 413
tflite::LocalResponseNormalizationOptionsBuilder, 414
tflite::LogicalAndOptionsBuilder, 415
tflite::LogicalNotOptionsBuilder, 415
tflite::LogicalOrOptionsBuilder, 416
tflite::LogSoftmaxOptionsBuilder, 416
tflite::LSHProjectionOptionsBuilder, 423
tflite::LSTMOptionsBuilder, 423
tflite::MatrixDiagOptionsBuilder, 424
tflite::MatrixSetDiagOptionsBuilder, 424
tflite::MaximumMinimumOptionsBuilder, 425
tflite::MetadataBuilder, 440
tflite::MirrorPadOptionsBuilder, 441
tflite::ModelBuilder, 441
tflite::MulOptionsBuilder, 446
tflite::NegOptionsBuilder, 452
tflite::NonMaxSuppressionV4OptionsBuilder, 453
tflite::NonMaxSuppressionV5OptionsBuilder, 453
tflite::NotEqualOptionsBuilder, 457
tflite::OneHotOptionsBuilder, 461
tflite::OperatorBuilder, 462
tflite::OperatorCodeBuilder, 463
tflite::PackOptionsBuilder, 477
tflite::PadOptionsBuilder, 485
tflite::PadV2OptionsBuilder, 485
tflite::Pool2DOptionsBuilder, 506
tflite::PowOptionsBuilder, 511
tflite::QuantizationDetailsTraits< T >, 545
tflite::QuantizationDetailsTraits< tflite::CustomQuantization >, 545
tflite::QuantizationParametersBuilder, 546
tflite::QuantizeOptionsBuilder, 546
tflite::RangeOptionsBuilder, 553
tflite::RankOptionsBuilder, 554
tflite::ReducerOptionsBuilder, 569
tflite::ReshapeOptionsBuilder, 575
tflite::ResizeBilinearOptionsBuilder, 576
tflite::ResizeNearestNeighborOptionsBuilder, 577
tflite::ReverseSequenceOptionsBuilder, 582
tflite::ReverseV2OptionsBuilder, 583
tflite::RNNOptionsBuilder, 583
tflite::ScatterNdOptionsBuilder, 616
tflite::SegmentSumOptionsBuilder, 619
tflite::SelectOptionsBuilder, 619
tflite::SelectV2OptionsBuilder, 620
tflite::SequenceRNNOptionsBuilder, 621
tflite::ShapeOptionsBuilder, 621
tflite::SignatureDefBuilder, 624
tflite::SkipGramOptionsBuilder, 624
tflite::SliceOptionsBuilder, 625
tflite::SoftmaxOptionsBuilder, 628
tflite::SpaceToBatchNDOptionsBuilder, 629
tflite::SpaceToDepthOptionsBuilder, 630
tflite::SparseIndexVectorTraits< T >, 630
tflite::SparseIndexVectorTraits< tflite::Int32Vector >, 631
tflite::SparseIndexVectorTraits< tflite::Uint16Vector >, 631

tflite::SparseIndexVectorTraits< tflite::Uint8Vector >, 631
tflite::SparseToDenseOptionsBuilder, 632
tflite::SparsityParametersBuilder, 632
tflite::SplitOptionsBuilder, 637
tflite::SplitVOptionsBuilder, 637
tflite::SquaredDifferenceOptionsBuilder, 638
tflite::SquareOptionsBuilder, 638
tflite::SqueezeOptionsBuilder, 639
tflite::StridedSliceOptionsBuilder, 644
tflite::SubGraphBuilder, 644
tflite::SubOptionsBuilder, 645
tflite::SVDFOptionsBuilder, 645
tflite::TensorBuilder, 665
tflite::TensorMapBuilder, 666
tflite::TileOptionsBuilder, 684
tflite::TopKV2OptionsBuilder, 686
tflite::TransposeConvOptionsBuilder, 689
tflite::TransposeOptionsBuilder, 689
tflite::Uint16VectorBuilder, 690
tflite::Uint8VectorBuilder, 690
tflite::UnidirectionalSequenceLSTMOptionsBuilder, 691
tflite::UniqueOptionsBuilder, 692
tflite::UnpackOptionsBuilder, 694
tflite::WhereOptionsBuilder, 734
tflite::WhileOptionsBuilder, 735
tflite::ZerosLikeOptionsBuilder, 737
TfliteInterpreter
nntrainer::TfliteInterpreter, 671
TfOpNode
nntrainer::TfOpNode, 674
toString
nntrainer::Connection, 196
Trainable
nntrainer::props::Trainable, 688
transposeloop
tensor.cpp, 963

UNIQUE
nntrainer::TensorSpecV2, 667
UNKNOWN
nntrainer, 64
unknownFactory
nntrainer::AppContext, 94
unloadExec
nntrainer::CachePool, 179
UNMANAGED
nntrainer, 63
unsetWorkingDirectory
nntrainer::AppContext, 94
unsubscribe
nntrainer::profile::Profiler, 526
updateTensor
nntrainer::RunLayerContext, 608
utils/base_properties.cpp, 974
utils/base_properties.h, 975
utils/ini_wrapper.cpp, 976
utils/ini_wrapper.h, 977
utils/nntr_threads.cpp, 978
utils/nntr_threads.h, 979
utils/node_exporter.cpp, 980
utils/node_exporter.h, 981
utils/profiler.cpp, 983
utils/profiler.h, 984
utils/tracer.h, 986
utils/util_func.cpp, 987
utils/util_func.h, 989

VALIDATE
nntrainer, 62
validate
nntrainer::CachePool, 179
nntrainer::InitLayerContext, 340
nntrainer::RunLayerContext, 608
validateIntervalOverlap
nntrainer, 75
value
nntrainer, 60, 62
value_type
nntrainer::GraphNodeIterator< LayerNodeType, GraphNodeType >, 303
var
nntrainer::Var_Grad, 709
Var_Grad
nntrainer::Var_Grad, 698–700
VarGradSpec
nntrainer, 58
VarGradSpecV2
nntrainer::VarGradSpecV2, 710, 711
variable_spec
nntrainer::VarGradSpecV2, 711
vg_spec
nntrainer::WeightSpecV2, 733

waitAndPop
nntrainer::ViewQueue< T >, 714
Weight
nntrainer::Weight, 717–719
WeightDecay
nntrainer::props::WeightDecay, 726
weightHasGradient
nntrainer::RunLayerContext, 609
WeightRegularizer
nntrainer, 64
WeightRegularizerConstant
nntrainer::props::WeightRegularizerConstant, 731
weightReorder
nntrainer::TfOpNode, 683
WeightSpec
nntrainer, 58
worker
nntrainer::TaskExecutor, 664
WRITE_BACK
nntrainer, 60