TIML

Generated by Doxygen 1.8.6

Fri Mar 20 2015 11:26:10

Contents

1 Introduction **1**

1.1 Overview . . . 1

1.2 Installation . . . 1

1.2.1 Dependencies . . . 1

1.2.2 Directory Structure . . . 2

1.2.3 Document Genearation . . . 3

1.2.4 Image Databases . . . 3

1.2.5 CNN Pretrained Models . . . 4

1.3 Convolutional Neural Networks (CNNs) . . . 6

1.3.1 Training Parameters . . . 6

1.3.2 Layers . . . 6

1.3.2.1 Input . . . 7

1.3.2.2 Convolutional . . . 7

1.3.2.3 Nonlinear . . . 7

1.3.2.4 Pooling . . . 7

1.3.2.5 Normalization . . . 7

1.3.2.6 Dropout . . . 7

1.3.2.7 Linear . . . 8

1.3.3 Memory . . . 8

1.3.4 Utility Functions . . . 8

1.3.5 Training . . . 10

1.3.6 Testing . . . 10

1.4 Applications . . . 10

1.4.1 Classification . . . 11

1.4.2 Scene Labeling . . . 12

1.5 Benchmarks . . . 14

2 Module Index **15**

2.1 Modules . . . 15

3 Data Structure Index **17**

3.1 Data Structures . . . 17

4 File Index 19
4.1 File List . . . 19
5 Module Documentation 25
5.1 appCNNClass . . . 25
5.1.1 Detailed Description . . . 25
5.2 appCNNConvertImageNet . . . 26
5.2.1 Detailed Description . . . 26
5.2.2 Function Documentation . . . 26
5.2.2.1 appCNNConvertImageNetShuffle . . . 26
5.2.2.2 main . . . 26
5.3 appCNNConvertSBD . . . 27
5.3.1 Detailed Description . . . 27
5.3.2 Function Documentation . . . 27
5.3.2.1 appCNNConvertSBDShuffle . . . 27
5.3.2.2 main . . . 27
5.4 appCNNInteropCaffe . . . 28
5.4.1 Detailed Description . . . 29
5.4.2 Function Documentation . . . 29
5.4.2.1 appCNNInteropCaffeConvert . . . 29
5.4.2.2 appCNNInteropCaffeConvLayerConvert . . . 29
5.4.2.3 appCNNInteropCaffeConvLayerPermuteKernel . . . 29
5.4.2.4 appCNNInteropCaffeDropoutLayerConvert . . . 29
5.4.2.5 appCNNInteropCaffeFillBlockDiagonalMatrix . . . 30
5.4.2.6 appCNNInteropCaffeFlipKernelMatrix . . . 30
5.4.2.7 appCNNInteropCaffeFlipMatrixFloat . . . 30
5.4.2.8 appCNNInteropCaffeLayerTypeConvert . . . 31
5.4.2.9 appCNNInteropCaffeLinearLayerConvert . . . 32
5.4.2.10 appCNNInteropCaffeNonlinearLayerConvert . . . 32
5.4.2.11 appCNNInteropCaffeNonlinearTypeConvert . . . 32
5.4.2.12 appCNNInteropCaffeNormLayerConvert . . . 32
5.4.2.13 appCNNInteropCaffePermuteMean . . . 33
5.4.2.14 appCNNInteropCaffePoolingLayerConvert . . . 33
5.4.2.15 appCNNInteropCaffePoolingTypeConvert . . . 33
5.4.2.16 appCNNInteropCaffeReadMean . . . 33
5.4.2.17 appCNNInteropCaffeReadProtoFromBinaryFile . . . 34
5.4.2.18 appCNNInteropCaffeReadProtoFromTextFile . . . 34
5.4.2.19 main . . . 34
5.5 appCNNScene . . . 35
5.5.1 Detailed Description . . . 35

5.5.2 Function Documentation . . . 35
5.5.2.1 appCNNSceneAccuracy . . . 35
5.5.2.2 appCNNSceneClassify . . . 36
5.5.2.3 appCNNSceneClassifyOpenMP . . . 36
5.5.2.4 appCNNSceneGetLabel . . . 36
5.5.2.5 appCNNSceneGetPatch . . . 37
5.5.2.6 appCNNSceneLabelMatrix . . . 38
5.5.2.7 appCNNSceneShuffleIdx . . . 38
5.5.2.8 appCNNSceneSupervisedTraining . . . 38
5.6 benchmarkCNNClass . . . 40
5.6.1 Detailed Description . . . 40
5.7 cnn . . . 41
5.7.1 Detailed Description . . . 46
5.7.2 Function Documentation . . . 46
5.7.2.1 timlCNNAddConvLayer . . . 46
5.7.2.2 timlCNNAddDropoutLayer . . . 46
5.7.2.3 timlCNNAddInputLayer . . . 47
5.7.2.4 timlCNNAddLinearLayer . . . 47
5.7.2.5 timlCNNAddNonlinearLayer . . . 47
5.7.2.6 timlCNNAddNormLayer . . . 47
5.7.2.7 timlCNNAddPoolingLayer . . . 48
5.7.2.8 timlCNNAssignDevice . . . 48
5.7.2.9 timlCNNBackPropagation . . . 48
5.7.2.10 timlCNNClassifyTop1SingleMode . . . 48
5.7.2.11 timlCNNClassifyTopNBatchMode . . . 49
5.7.2.12 timlCNNClassifyTopNBatchModeOpenMP . . . 49
5.7.2.13 timlCNNClassifyTopNTeamModeOpenMP . . . 49
5.7.2.14 timlCNNClone . . . 50
5.7.2.15 timlCNNConvBackPropagation . . . 50
5.7.2.16 timlCNNConvForwardPropagation . . . 50
5.7.2.17 timlCNNConvInitialize . . . 50
5.7.2.18 timlCNNConvParamsDefault . . . 51
5.7.2.19 timlCNNConvReadFromBinaryFile . . . 51
5.7.2.20 timlCNNConvReadFromTextFile . . . 51
5.7.2.21 timlCNNConvShareParams . . . 51
5.7.2.22 timlCNNConvUpdateParams . . . 52
5.7.2.23 timlCNNConvWriteToFile . . . 52
5.7.2.24 timlCNNCostWithLabel . . . 52
5.7.2.25 timlCNNCreateConvNeuralNetwork . . . 53
5.7.2.26 timlCNNDelete . . . 54

5.7.2.27 timlCNNDeleteConvLayer 54
5.7.2.28 timlCNNDeleteDropoutLayer 54
5.7.2.29 timlCNNDeleteInputLayer 54
5.7.2.30 timlCNNDeleteLinearLayer 55
5.7.2.31 timlCNNDeleteNonlinearLayer 56
5.7.2.32 timlCNNDeleteNormLayer 56
5.7.2.33 timlCNNDeletePoolingLayer 56
5.7.2.34 timlCNNDropoutBackPropagation 56
5.7.2.35 timlCNNDropoutForwardPropagation 57
5.7.2.36 timlCNNDropoutInitialize 58
5.7.2.37 timlCNNDropoutReadFromTextFile 58
5.7.2.38 timlCNNDropoutWriteToFile 58
5.7.2.39 timlCNNForwardPropagation 58
5.7.2.40 timlCNNGetLayerNum 59
5.7.2.41 timlCNNGetParamsNum 59
5.7.2.42 timlCNNInitialize 59
5.7.2.43 timlCNNInputForwardPropagation 59
5.7.2.44 timlCNNInputInitialize 60
5.7.2.45 timlCNNInputParamsDefault 61
5.7.2.46 timlCNNInputReadFromBinaryFile 61
5.7.2.47 timlCNNInputReadFromTextFile 61
5.7.2.48 timlCNNInputShareParams 61
5.7.2.49 timlCNNInputWriteToFile 62
5.7.2.50 timlCNNLayerTypeStr 62
5.7.2.51 timlCNNLinearBackPropagation 62
5.7.2.52 timlCNNLinearForwardPropagation 62
5.7.2.53 timlCNNLinearInitialize 63
5.7.2.54 timlCNNLinearParamsDefault 63
5.7.2.55 timlCNNLinearReadFromBinaryFile 63
5.7.2.56 timlCNNLinearReadFromTextFile 63
5.7.2.57 timlCNNLinearShareParams 64
5.7.2.58 timlCNNLinearUpdateParams 65
5.7.2.59 timlCNNLinearWriteToFile 65
5.7.2.60 timlCNNMaxPoolingBackPropagation 65
5.7.2.61 timlCNNMaxPoolingForwardPropagation 65
5.7.2.62 timlCNNMeanPoolingBackPropagation 66
5.7.2.63 timlCNNMeanPoolingForwardPropagation 66
5.7.2.64 timlCNNMemory 66
5.7.2.65 timlCNNMemPoolSize 66
5.7.2.66 timlCNNNonlinearBackPropagation 66

5.7.2.67 timlCNNNonlinearForwardPropagation 67
5.7.2.68 timlCNNNonlinearInitialize 67
5.7.2.69 timlCNNNonlinearParamsDefault 67
5.7.2.70 timlCNNNonlinearReadFromTextFile 67
5.7.2.71 timlCNNNonlinearWriteToFile 67
5.7.2.72 timlCNNNormBackPropagation 68
5.7.2.73 timlCNNNormForwardPropagation 68
5.7.2.74 timlCNNNormInitialize 68
5.7.2.75 timlCNNNormParamsDefault 68
5.7.2.76 timlCNNNormReadFromTextFile 69
5.7.2.77 timlCNNNormWriteToFile 69
5.7.2.78 timlCNNPoolingBackPropagation 69
5.7.2.79 timlCNNPoolingForwardPropagation 69
5.7.2.80 timlCNNPoolingInitialize 70
5.7.2.81 timlCNNPoolingParamsDefault 71
5.7.2.82 timlCNNPoolingReadFromTextFile 71
5.7.2.83 timlCNNPoolingWriteToFile 71
5.7.2.84 timlCNNPrint 71
5.7.2.85 timlCNNProfile 72
5.7.2.86 timlCNNReadFromFile 72
5.7.2.87 timlCNNReset 72
5.7.2.88 timlCNNResetConvLayer 72
5.7.2.89 timlCNNResetDropoutLayer 73
5.7.2.90 timlCNNResetInputLayer 73
5.7.2.91 timlCNNResetLinearLayer 73
5.7.2.92 timlCNNResetNonlinearLayer 73
5.7.2.93 timlCNNResetNormLayer 74
5.7.2.94 timlCNNResetPoolingLayer 75
5.7.2.95 timlCNNResize 75
5.7.2.96 timlCNNSetMode 75
5.7.2.97 timlCNNShareParams 75
5.7.2.98 timlCNNSupervisedTrainingWithLabelBatchMode 76
5.7.2.99 timlCNNSupervisedTrainingWithLabelBatchModeOpenMP 76
5.7.2.100 timlCNNTrainingParamsDefault 76
5.7.2.101 timlCNNTrainingParamsReadFromTextFile 76
5.7.2.102 timlCNNTrainingParamsWriteToFile 77
5.7.2.103 timlCNNUpdateParams 77
5.7.2.104 timlCNNWriteToFile 77
5.8 util 78
5.8.1 Detailed Description 81

5.8.2 Enumeration Type Documentation 81
5.8.2.1 timlUtilAllocatorLevel 81
5.8.2.2 timlUtilCropType 82
5.8.2.3 timlUtilMirrorType 82
5.8.2.4 timlUtilParamsLevel 82
5.8.3 Function Documentation 82
5.8.3.1 timlUtilClassifyAccuracy 82
5.8.3.2 timlUtilConv2Full 82
5.8.3.3 timlUtilConv2ImageReshape 83
5.8.3.4 timlUtilConv2ImageReshapeBack 83
5.8.3.5 timlUtilConv2ImageReshapeIndex 83
5.8.3.6 timlUtilConv2Valid 84
5.8.3.7 timlUtilCorr2Full 84
5.8.3.8 timlUtilDiffTime 85
5.8.3.9 timlUtilElementWiseFunction 86
5.8.3.10 timlUtilElementWiseMultiply 86
5.8.3.11 timlUtilFread 86
5.8.3.12 timlUtilFree 86
5.8.3.13 timlUtilFwrite 87
5.8.3.14 timlUtilLocalContrastNormalize 87
5.8.3.15 timlUtilLocalContrastUnnormalize 87
5.8.3.16 timlUtilMalloc 88
5.8.3.17 timlUtilMasking 88
5.8.3.18 timlUtilMaxPooling 88
5.8.3.19 timlUtilMeanPooling 89
5.8.3.20 timlUtilMeanSqaureError 89
5.8.3.21 timlUtilMultinomialCrossEntropy 90
5.8.3.22 timlUtilRandContinuousUniformRNG 90
5.8.3.23 timlUtilRandDiscreteUniformRNG 90
5.8.3.24 timlUtilRandNormalRNG 90
5.8.3.25 timlUtilRandPerm 91
5.8.3.26 timlUtilReadCIFAR10 91
5.8.3.27 timlUtilReadFixedSizeJPEG 91
5.8.3.28 timlUtilReadJPEG 91
5.8.3.29 timlUtilReadMNIST 92
5.8.3.30 timlUtilRelu 92
5.8.3.31 timlUtilReluDerivative 92
5.8.3.32 timlUtilReverseEndian32 92
5.8.3.33 timlUtilScanJPEG 93
5.8.3.34 timlUtilSigmoid 93

5.8.3.35 timlUtilSigmoidDerivative 93
5.8.3.36 timlUtilSoftmax 93
5.8.3.37 timlUtilSubtract 94
5.8.3.38 timlUtilTanh 94
5.8.3.39 timlUtilTanhDerivative 94
5.8.3.40 timlUtilTransform 95
5.8.3.41 timlUtilUndoMaxPooling 95
5.8.3.42 timlUtilUndoMeanPooling 95
5.8.3.43 timlUtilUnmasking 96
5.8.3.44 timlUtilVectorMaxFloat 96
5.8.3.45 timlUtilVectorMaxIndexFloat 97
5.8.3.46 timlUtilVectorResetFloat 98
5.8.3.47 timlUtilVectorResetInt 98
5.8.3.48 timlUtilVectorSortFloat 98
5.8.3.49 timlUtilVectorSortIndexFloat 98
5.8.3.50 timlUtilVectorSumFloat 99
5.9 testCNN 100
5.9.1 Detailed Description 100
5.9.2 Function Documentation 100
5.9.2.1 testCNNSimpleClone 100
5.9.2.2 testCNNSimpleIO 100
5.9.2.3 testCNNSimpleProfile 100
5.9.2.4 testCNNSimpleResize 101
5.9.2.5 testCNNSimpleShare 101
5.9.2.6 testCNNSimpleTraining 101
5.10 testUtil 102
5.10.1 Detailed Description 102
5.10.2 Function Documentation 102
5.10.2.1 testUtilBLAS 102
5.10.2.2 testUtilConv2 102
5.10.2.3 testUtilSort 102
5.11 app 103
5.11.1 Detailed Description 103
5.12 appCNN 104
5.12.1 Detailed Description 104
5.13 benchmark 105
5.13.1 Detailed Description 105
5.14 benchmarkCNN 106
5.14.1 Detailed Description 106
5.15 test 107

5.15.1 Detailed Description 107
5.15.2 Function Documentation 107
5.15.2.1 testCNNSimpleProfile 107

6 Data Structure Documentation 109

6.1 _timlCNNLayer_ Struct Reference 109
6.1.1 Field Documentation 109
6.1.1.1 delta 109
6.1.1.2 dropoutParams 109
6.1.1.3 prev 110
6.2 _timlConvNeuralNetwork_ Struct Reference 110
6.2.1 Field Documentation 110
6.2.1.1 memPool 110
6.2.1.2 memPoolSize 110
6.3 appCNNSceneDataSet Struct Reference 110
6.3.1 Field Documentation 110
6.3.1.1 patchSize 110
6.4 timlCNNConvParams Struct Reference 111
6.4.1 Field Documentation 111
6.4.1.1 connectivity 111
6.4.1.2 kernel 111
6.4.1.3 prevFeatureMapReshape 111
6.4.1.4 prevFeatureMapReshapeIndex 112
6.5 timlCNNDataSet Struct Reference 112
6.5.1 Field Documentation 112
6.5.1.1 data 112
6.5.1.2 label 112
6.6 timlCNNDropoutParams Struct Reference 112
6.6.1 Field Documentation 112
6.6.1.1 mask 112
6.6.1.2 prob 112
6.6.1.3 randomVector 113
6.7 timlCNNInputParams Struct Reference 113
6.7.1 Field Documentation 113
6.7.1.1 channel 113
6.7.1.2 channelPermute 113
6.7.1.3 col 113
6.7.1.4 inputData 113
6.7.1.5 mean 113
6.7.1.6 row 113

6.7.1.7 shared 114
6.8 timlCNNLinearParams Struct Reference 114
6.8.1 Field Documentation 114
6.8.1.1 dim 114
6.8.1.2 prevDim 114
6.8.1.3 shared 114
6.9 timlCNNNonlinearParams Struct Reference 114
6.10 timlCNNNormParams Struct Reference 115
6.10.1 Field Documentation 115
6.10.1.1 denom 115
6.11 timlCNNPoolingParams Struct Reference 115
6.11.1 Field Documentation 115
6.11.1.1 maxIndex 115
6.11.1.2 scaleCol 115
6.11.1.3 scaleRow 115
6.11.1.4 strideX 116
6.11.1.5 strideY 116
6.12 timlCNNTrainingParams Struct Reference 116
6.12.1 Field Documentation 116
6.12.1.1 batchCount 116
6.12.1.2 batchSize 116
6.12.1.3 costType 116
6.12.1.4 count 116
6.12.1.5 epoch 116
6.13 timlUtilImage Struct Reference 117
6.14 timlUtilImageSet Struct Reference 117
6.14.1 Field Documentation 117
6.14.1.1 mean 117
6.14.1.2 num 117
6.15 timlUtilInitializer Struct Reference 117
6.15.1 Field Documentation 118
6.15.1.1 max 118
6.15.1.2 mean 118
6.15.1.3 min 118
6.15.1.4 std 118
6.15.1.5 val 118

7 File Documentation 119

7.1 appCNNClass.h File Reference 119
7.2 appCNNClassCIFAR10Testing.c File Reference 119

7.3 appCNNClassCIFAR10Training.c File Reference . . . 120
7.4 appCNNClassImageNetAlexNetTesting.c File Reference . . . 120
7.5 appCNNClassImageNetCaffeNetTesting.c File Reference . . . 121
7.6 appCNNClassImageNetCaffeNetTraining.c File Reference . . . 121
7.7 appCNNClassImageNetVGGNetTesting.c File Reference . . . 121
7.8 appCNNClassMNISTTesting.c File Reference . . . 122
7.9 appCNNClassMNISTTraining.c File Reference . . . 122
7.10 appCNNConvertImageNet.cpp File Reference . . . 123
7.11 appCNNConvertImageNet.hpp File Reference . . . 123
7.12 appCNNConvertSBD.cpp File Reference . . . 124
7.13 appCNNConvertSBD.hpp File Reference . . . 124
7.14 appCNNConvertSBDShuffle.cpp File Reference . . . 124
7.15 appCNNInteropCaffe.cpp File Reference . . . 125
7.16 appCNNInteropCaffe.hpp File Reference . . . 125
7.17 appCNNInteropCaffeConvert.cpp File Reference . . . 126
7.18 appCNNInteropCaffeConvLayerConvert.cpp File Reference . . . 126
7.19 appCNNInteropCaffeConvLayerPermuteKernel.cpp File Reference . . . 127
7.20 appCNNInteropCaffeDropoutLayerConvert.cpp File Reference . . . 127
7.21 appCNNInteropCaffeFillBlockDiagonalMatrix.cpp File Reference . . . 127
7.22 appCNNInteropCaffeFlipKernelMatrix.cpp File Reference . . . 127
7.23 appCNNInteropCaffeFlipMatrixFloat.cpp File Reference . . . 128
7.24 appCNNInteropCaffeLayerTypeConvert.cpp File Reference . . . 128
7.25 appCNNInteropCaffeLinearLayerConvert.cpp File Reference . . . 128
7.26 appCNNInteropCaffeNonlinearLayerConvert.cpp File Reference . . . 128
7.27 appCNNInteropCaffeNonlinearTypeConvert.cpp File Reference . . . 128
7.28 appCNNInteropCaffeNormLayerConvert.cpp File Reference . . . 129
7.29 appCNNInteropCaffePermuteMean.cpp File Reference . . . 129
7.30 appCNNInteropCaffePoolingLayerConvert.cpp File Reference . . . 129
7.31 appCNNInteropCaffePoolingTypeConvert.cpp File Reference . . . 129
7.32 appCNNInteropCaffeReadMean.cpp File Reference . . . 130
7.33 appCNNInteropCaffeReadProtoFromBinaryFile.cpp File Reference . . . 130
7.34 appCNNInteropCaffeReadProtoFromTextFile.cpp File Reference . . . 130
7.35 appCNNScene.h File Reference . . . 130
7.36 appCNNSceneAccuracy.c File Reference . . . 131
7.37 appCNNSceneClassify.c File Reference . . . 131
7.38 appCNNSceneClassifyOpenMP.c File Reference . . . 131
7.39 appCNNSceneGetLabel.c File Reference . . . 132
7.40 appCNNSceneGetPatch.c File Reference . . . 132
7.41 appCNNSceneLabelMatrix.c File Reference . . . 132
7.42 appCNNSceneSBDTesting.c File Reference . . . 132

7.43 appCNNSceneSBDTraining.c File Reference 133
7.44 appCNNSceneShuffleIdx.c File Reference 133
7.45 appCNNSceneSupervisedTraining.c File Reference 133
7.46 benchmarkCNNClass.h File Reference 134
7.47 benchmarkCNNClassCaffeNetTesting.c File Reference 134
7.48 benchmarkCNNClassVGGNetTesting.c File Reference 134
7.49 testCNN.c File Reference 135
7.50 testCNN.h File Reference 135
7.51 testCNNSimpleClone.c File Reference 135
7.52 testCNNSimpleIO.c File Reference 136
7.53 testCNNSimpleProfile.c File Reference 136
7.54 testCNNSimpleResize.c File Reference 137
7.55 testCNNSimpleShare.c File Reference 137
7.56 testCNNSimpleTraining.c File Reference 138
7.57 testUtil.c File Reference 138
7.58 testUtil.h File Reference 138
7.59 testUtilBLAS.c File Reference 138
7.60 testUtilConv2.c File Reference 139
7.61 testUtilSort.c File Reference 139
7.62 timl.h File Reference 139
7.62.1 Detailed Description 141
7.63 timlCNN.h File Reference 141
7.64 timlCNNAddConvLayer.c File Reference 145
7.65 timlCNNAddDropoutLayer.c File Reference 145
7.66 timlCNNAddInputLayer.c File Reference 145
7.67 timlCNNAddLinearLayer.c File Reference 145
7.68 timlCNNAddNonlinearLayer.c File Reference 146
7.69 timlCNNAddNormLayer.c File Reference 146
7.70 timlCNNAddPoolingLayer.c File Reference 146
7.71 timlCNNAssignDevice.c File Reference 146
7.72 timlCNNBackPropagation.c File Reference 147
7.73 timlCNNClassifyTop1SingleMode.c File Reference 147
7.74 timlCNNClassifyTopNBatchMode.c File Reference 147
7.75 timlCNNClassifyTopNBatchModeOpenMP.c File Reference 147
7.76 timlCNNClassifyTopNTeamModeOpenMP.c File Reference 147
7.77 timlCNNClone.c File Reference 148
7.78 timlCNNConvBackPropagation.c File Reference 148
7.79 timlCNNConvForwardPropagation.c File Reference 148
7.80 timlCNNConvInitialize.c File Reference 148
7.81 timlCNNConvParamsDefault.c File Reference 149

7.82 timlCNNConvReadFromBinaryFile.c File Reference . . . 149
7.83 timlCNNConvReadFromTextFile.c File Reference . . . 149
7.84 timlCNNConvShareParams.c File Reference . . . 149
7.85 timlCNNConvUpdateParams.c File Reference . . . 149
7.86 timlCNNConvWriteToFile.c File Reference . . . 150
7.87 timlCNNCostWithLabel.c File Reference . . . 150
7.88 timlCNNCreateConvNeuralNetwork.c File Reference . . . 150
7.89 timlCNNDelete.c File Reference . . . 150
7.90 timlCNNDeleteConvLayer.c File Reference . . . 151
7.91 timlCNNDeleteDropoutLayer.c File Reference . . . 151
7.92 timlCNNDeleteInputLayer.c File Reference . . . 151
7.93 timlCNNDeleteLinearLayer.c File Reference . . . 151
7.94 timlCNNDeleteNonlinearLayer.c File Reference . . . 151
7.95 timlCNNDeleteNormLayer.c File Reference . . . 152
7.96 timlCNNDeletePoolingLayer.c File Reference . . . 152
7.97 timlCNNDropoutBackPropagation.c File Reference . . . 152
7.98 timlCNNDropoutForwardPropagation.c File Reference . . . 152
7.99 timlCNNDropoutInitialize.c File Reference . . . 153
7.100timlCNNDropoutReadFromTextFile.c File Reference . . . 153
7.101timlCNNDropoutWriteToFile.c File Reference . . . 153
7.102timlCNNForwardPropagation.c File Reference . . . 153
7.103timlCNNGetLayerNum.c File Reference . . . 153
7.104timlCNNGetParamsNum.c File Reference . . . 154
7.105timlCNNInitialize.c File Reference . . . 154
7.106timlCNNInputInitialize.c File Reference . . . 154
7.107timlCNNInputParamsDefault.c File Reference . . . 154
7.108timlCNNInputReadFromBinaryFile.c File Reference . . . 155
7.109timlCNNInputReadFromTextFile.c File Reference . . . 155
7.110timlCNNInputShareParams.c File Reference . . . 155
7.111timlCNNInputWriteToFile.c File Reference . . . 155
7.112timlCNNLayerTypeStr.c File Reference . . . 155
7.113timlCNNLinearBackPropagation.c File Reference . . . 156
7.114timlCNNLinearForwardPropagation.c File Reference . . . 156
7.115timlCNNLinearInitialize.c File Reference . . . 156
7.116timlCNNLinearParamsDefault.c File Reference . . . 156
7.117timlCNNLinearReadFromBinaryFile.c File Reference . . . 157
7.118timlCNNLinearReadFromTextFile.c File Reference . . . 157
7.119timlCNNLinearShareParams.c File Reference . . . 157
7.120timlCNNLinearUpdateParams.c File Reference . . . 157
7.121timlCNNLinearWriteToFile.c File Reference . . . 157

7.122timlCNNMaxPoolingBackPropagation.c File Reference . . . 158
7.123timlCNNMaxPoolingForwardPropagation.c File Reference . . . 158
7.124timlCNNMeanPoolingBackPropagation.c File Reference . . . 158
7.125timlCNNMeanPoolingForwardPropagation.c File Reference . . . 158
7.126timlCNNMemory.c File Reference . . . 159
7.127timlCNNMemPoolSize.c File Reference . . . 159
7.128timlCNNNonlinearBackPropagation.c File Reference . . . 159
7.129timlCNNNonlinearForwardPropagation.c File Reference . . . 159
7.130timlCNNNonlinearInitialize.c File Reference . . . 159
7.131timlCNNNonlinearParamsDefault.c File Reference . . . 160
7.132timlCNNNonlinearReadFromTextFile.c File Reference . . . 160
7.133timlCNNNonlinearWriteToFile.c File Reference . . . 160
7.134timlCNNNormBackPropagation.c File Reference . . . 160
7.135timlCNNNormForwardPropagation.c File Reference . . . 161
7.136timlCNNNormInitialize.c File Reference . . . 161
7.137timlCNNNormParamsDefault.c File Reference . . . 161
7.138timlCNNNormWriteToFile.c File Reference . . . 161
7.139timlCNNPoolingBackPropagation.c File Reference . . . 161
7.140timlCNNPoolingForwardPropagation.c File Reference . . . 162
7.141timlCNNPoolingInitialize.c File Reference . . . 162
7.142timlCNNPoolingParamsDefault.c File Reference . . . 162
7.143timlCNNPoolingReadFromTextFile.c File Reference . . . 162
7.144timlCNNPoolingWriteToFile.c File Reference . . . 163
7.145timlCNNPrint.c File Reference . . . 163
7.146timlCNNProfile.c File Reference . . . 163
7.147timlCNNReadFromFile.c File Reference . . . 163
7.148timlCNNReset.c File Reference . . . 163
7.149timlCNNResetConvLayer.c File Reference . . . 164
7.150timlCNNResetDropoutLayer.c File Reference . . . 164
7.151timlCNNResetInputLayer.c File Reference . . . 164
7.152timlCNNResetLinearLayer.c File Reference . . . 164
7.153timlCNNResetNonlinearLayer.c File Reference . . . 165
7.154timlCNNResetNormLayer.c File Reference . . . 165
7.155timlCNNResetPoolingLayer.c File Reference . . . 165
7.156timlCNNResize.c File Reference . . . 165
7.157timlCNNSetMode.c File Reference . . . 165
7.158timlCNNShareParams.c File Reference . . . 166
7.159timlCNNSupervisedTrainingWithLabelBatchMode.c File Reference . . . 166
7.160timlCNNSupervisedTrainingWithLabelBatchModeOpenMP.c File Reference . . . 166
7.161timlCNNTrainingParamsDefault.c File Reference . . . 166

7.162timlCNNTrainingParamsReadFromTextFile.c File Reference 167
7.163timlCNNTrainingParamsWriteToFile.c File Reference 167
7.164timlCNNUpdateParams.c File Reference 167
7.165timlCNNWriteToFile.c File Reference 167
7.166timlUtil.h File Reference 168
7.167timlUtilBLAS.c File Reference 172
7.168timlUtilClassifyAccuracy.c File Reference 172
7.169timlUtilConv2Full.c File Reference 173
7.170timlUtilConv2ImageReshape.c File Reference 173
7.171timlUtilConv2ImageReshapeBack.c File Reference 173
7.172timlUtilConv2ImageReshapeIndex.c File Reference 173
7.173timlUtilConv2Valid.c File Reference 173
7.174timlUtilCorr2Full.c File Reference 174
7.175timlUtilDiffTime.c File Reference 174
7.176timlUtilElementWiseFunction.c File Reference 174
7.177timlUtilElementWiseMultiply.c File Reference 174
7.178timlUtilFread.c File Reference 175
7.179timlUtilFree.c File Reference 175
7.180timlUtilFwrite.c File Reference 175
7.181timlUtilLocalContrastNormalize.c File Reference 175
7.182timlUtilLocalContrastUnnormalize.c File Reference 175
7.183timlUtilMalloc.c File Reference 176
7.184timlUtilMasking.c File Reference 176
7.185timlUtilMaxPooling.c File Reference 176
7.186timlUtilMeanPooling.c File Reference 176
7.187timlUtilMeanSquareError.c File Reference 177
7.188timlUtilMultinomialCrossEntropy.c File Reference 177
7.189timlUtilRandContinuousUniformRNG.c File Reference 177
7.190timlUtilRandDiscreteUniformRNG.c File Reference 177
7.191timlUtilRandNormalRNG.c File Reference 178
7.192timlUtilRandPerm.c File Reference 178
7.193timlUtilReadCIFAR10.c File Reference 178
7.194timlUtilReadFixedSizeJPEG.c File Reference 178
7.195timlUtilReadJPEG.c File Reference 179
7.196timlUtilReadMNIST.c File Reference 179
7.197timlUtilRelu.c File Reference 179
7.198timlUtilReluDerivative.c File Reference 179
7.199timlUtilReverseEndian32.c File Reference 180
7.200timlUtilScanJPEG.c File Reference 180
7.201timlUtilSigmoid.c File Reference 180

7.202timlUtilSigmoidDerivative.c File Reference 180
7.203timlUtilSoftmax.c File Reference 180
7.204timlUtilSubtract.c File Reference 181
7.205timlUtilTanh.c File Reference 181
7.206timlUtilTanhDerivative.c File Reference 181
7.207timlUtilTransform.c File Reference 181
7.208timlUtilUndoMaxPooling.c File Reference 182
7.209timlUtilUndoMeanPooling.c File Reference 182
7.210timlUtilUnmasking.c File Reference 182
7.211timlUtilVectorMaxFloat.c File Reference 182
7.212timlUtilVectorMaxIndexFloat.c File Reference 182
7.213timlUtilVectorResetFloat.c File Reference 183
7.214timlUtilVectorResetInt.c File Reference 183
7.215timlUtilVectorSortFloat.c File Reference 183
7.216timlUtilVectorSortIndexFloat.c File Reference 183
7.217timlUtilVectorSumFloat.c File Reference 184

Bibliography **185**

Index **186**

Chapter 1

Introduction

1.1 Overview

The Texas Instruments Machine Learning (TIML) library is C implementation of common machine learning functions optimized for TI embedded devices. Initially, the library supports convolutional neural networks (CNNs), but will grow with time in scope. The library has a minimal set of dependencies upon other libraries to simplify the installation and use.

1.2 Installation

1.2.1 Dependencies

TIML library dependencies

The TIML library requires a high performance C basic linear algebra subprogram (CBLAS) library and a library for reading Joint Photographic Experts Group (JPEG) images. An optimized CBLAS implementation is included with TI embedded devices. For pre development work on a desktop, a reference (unoptimized) CBLAS implementation can be downloaded from http://www.netlib.org/blas/blast-forum/cblas.tgz or in Ubuntu using the command

```
sudo apt-get install libatlas-base-dev
```

A JPEG library, libjpeg, can be downloaded from

http://sourceforge.net/projects/libjpeg/files/latest/download?source=files

or in Ubuntu using the command

```
sudo apt-get install libjpeg-dev
```

Application dependencies

The ImageNet database conversion application resizes all the images to size 256$*$256. The resizing is done using openCV. In Ubuntu, use the following command to install openCV

```
sudo apt-get install libopencv-dev
```

The interop application converts a CNN model in Caffe format to TIML CNN format. The Caffe format depends on google's protocol buffer library. To install the library, use command

```
sudo apt-get install libprotobuf-dev
```

1.2.2 Directory Structure

Install the TIML library by unzipping the zip file to a directory referred to as <install_directory>. At a terminal prompt, change the working directory to `<install_directory>/build` and use the command `make all` to compile the library. The library file will be placed inside the `<install_directory>/bin` folder.

The directory structure of the compiled library is shown as follows:

- `<install_directory>/bin`: library binary file directory
- `<install_directory>/build`: makefile directory
- `<install_directory>/doc`: doxygen file directory
- `<install_directory>/src/common`: common library directory
  - `<install_directory>/src/common/api`: library API source file
  - `<install_directory>/src/common/cnn`: cnn moudule source file
  - `<install_directory>/src/common/util`: util moudule source file
- `<install_directory>/src/test`: test module directory
  - `<install_directory>/src/test/cnn`: cnn test source file
  - `<install_directory>/src/test/util`: util test source file
- `<install_directory>/src/benchmark`: benchmark module directory
  - `<install_directory>/src/benchmark/cnn`: cnn benchmark source file
    - `<install_directory>/src/benchmark/cnn/class`: cnn classification benchmark source file
- `<install_directory>/src/app`: application module directory
  - `<install_directory>/src/app/cnn`: cnn application source file
    - `<install_directory>/src/app/cnn/class`: cnn classification application source file
      - `<install_directory>/src/app/cnn/class/cifar10`: cnn CIFAR10 database classification application source file
      - `<install_directory>/src/app/cnn/class/imagenet`: cnn ImageNet database classification application source file
      - `<install_directory>/src/app/cnn/class/mnist`: cnn MNIST database classification application source file
    - `<install_directory>/src/app/cnn/scene`: cnn scene labeling application source file
      - `<install_directory>/src/app/cnn/scene/sbd`: cnn stanford background dataset scene labeling application source file
    - `<install_directory>/src/app/cnn/interop`: cnn interoperation application source file
      - `<install_directory>/src/app/cnn/interop/caffe`: cnn-caffe interoperation application source file
    - `<install_directory>/src/app/cnn/convert`: cnn database conversion application source file
      - `<install_directory>/src/app/cnn/convert/imagenet`: cnn ImageNet database conversion application source file
      - `<install_directory>/src/app/cnn/convert/sbd`: cnn SBD database conversion application source file
- `<install_directory>/src/database`: database directory
  - cifar10: Canadian Institute for Advanced Research-10 Class
  - imagenet: ImageNet 2012

- mnist: Mixed National Institute of Standards and Technology
- sbd: Stanford Background Dataset
- model: pretrained models
  - cifar10
  - alexnet
  - caffenet
  - vggnet
  - mnist
  - sbd

1.2.3 Document Genearation

Documents of html and pdf formats can be generated using Doxygen. Make sure you have both latex and Doxygen installed. In Ubuntu, the installation commands are

- `sudo apt-get install doxygen`
- `sudo apt-get install texlive-full`

Use the command `doxygen <install- <install_directory>/src/app/cnn/scene: cnn scene labeling application source file_directory>/doc/timl.Doxyfile` to generate the documents. The html version is located at `doxygen <install_directory>/doc/html`. To generate the pdf version, change the directory to `doxygen <install_directory>/doc/latex` and run the command `make`. A filed named refman.pdf will be generated in the currnet folder.

1.2.4 Image Databases

In order to run the examples provided by the library, it is required to download additional databases of test images.

MNIST

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

You can download the MNIST database from `http://yann.lecun.com/exdb/mnist/` or simpliy change the directory to `<install_directory>/src/database/mnist` and run the script `./databaseMNISTDownload.sh`. After the download, there should be 4 files in the folder:

- `t10k-images.idx3-ubyte`
- `t10k-labels.idx1-ubyte`
- `train-images.idx3-ubyte`
- `train-labels.idx1-ubyte`.

CIFAR10

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

You can download the CIFAR10 database from `http://www.cs.toronto.edu/~kriz/cifar.html` or simpliy change the directory to `<install_directory>/src/database/cifar10` and run the script `./databaseCIFAR10Download.sh`. After the download, there should be 6 files in the folder:

- `data_batch_1.bin`
- `data_batch_2.bin`
- `data_batch_3.bin`
- `data_batch_4.bin`
- `data_batch_5.bin`
- `test_batch_6.bin`

ImageNet

ImageNet is an image dataset organized according to the WordNet hierarchy.Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set" or "synset". There are more than 100,000 synsets in WordNet, majority of them are nouns (80,000+). ImageNet aims to provide on average 1000 images to illustrate each synset. Images of each concept are quality-controlled and human-annotated.

Download the database from `http://www.image-net.org/challenges/LSVRC/2012/nonpub-downloads` to <install_directory>/database/imagnet. You need to register before you can download the database.http-://dags.stanford.edu/projects/scenedataset.html

- Download the training images of size 138GB and MD5: 1d675b47d978889d74fa0da5fadfb00e
- Download the validation images of size 6.3GB and MD5: 29b22e2961454d5413ddabcf34fc5622
- Download the auxiliary files by changing the working directory to `<install_directory>/src/database/imagnet` and runing the script `./databaseImageNetDownload.sh`
- Extract all the files to the <install_directory>/src/database/imagnet
- Change run the script ./databaseImageNetConvert.sh to convert the raw database to a format that is ready to be processed by TIML. You may need to change the path variables in this script if you choose to store the database in other directories. You can also specify the number of images to be converted in the script.

After the conversion, there should be two folders, test and train, in <install_directory>/src/database/imagnet. There will be a single labels.txt file in each of the folder. All the images will be resized to 256∗256.

Stanford Background Dataset

The Stanford Background Dataset is a new dataset introduced in Gould et al. (ICCV 2009) for evaluating methods for geometric and semantic scene understanding. The dataset contains 715 images chosen from existing public datasets: LabelMe, MSRC, PASCAL VOC and Geometric Context. The selection criteria were for the images to be of outdoor scenes, have approximately 320-by-240 pixels, contain at least one foreground object, and have the horizon position within the image.

You can download the database from `http://dags.stanford.edu/data/iccv09Data.tar.gz` or simply change the working directory to `<install_directory>/src/database/sbd` and run the script `databaseSBDDownload.sh` To convert the database to TIML compatible format, run the script ./databaseSBDConvert.sh. You can specify the number of images used for training and testing by changing the corresponding variables in the script. After running the script, there should be two folders named "train" and "test" in the current folder.

1.2.5 CNN Pretrained Models

TIML can convert models pretrained by Caffe[2] to a format that is compatible with TIML. Caffe is a deep learning framework developed by the Berkeley Vision and Learning Center (BVLC). You can train your CNN on Caffe on CPU or GPU, save the parameters and then convert it to a format supported by TIML.

AlexNet [1]

AlexNet is a deep convolutional neural network to classify the 1.3 million high-resolution images in the LSVRC-2010 ImageNet training set into the 1000 different classes. On the test data, AlexNet achieved top-1 and top-5 error rates of 39.7% and 18.9% which is considerably better than the previous state-of-the-art results. The neural network, which has 60 million parameters and 500,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and two globally connected layers with a final 1000-way softmax.

- Download the Caffe AlexNet binary file from http://dl.caffe.berkeleyvision.org/bvlc_alexnet.caffemodel to <install_directory>/src/database/model/alexnet
- Download the Caffe AlexNet depoly file from https://github.com/BVLC/caffe/blob/master/models/bvlc_alexnet/deploy.prototxt to <install_directory>/src/database/model/alexnet
- Run the script ./databaseModelAlexNetInterop.sh to perform the conversion

After the conversion, there should two files in <install_directory>/database/model/alexnet:

- databaseModelAlexNet.m *A text file that defines the structure of the CNN*
- databaseModelAlexNet.m.params *A binary file that stores the parameters of the CNN*

CaffeNet

The CaffeNet is a replication of the model described in the AlexNet publication with some differences:

- not training with the relighting data-augmentation;
- the order of pooling and normalization layers is switched (in CaffeNet, pooling is done before normalization).

This model obtains a top-1 accuracy 57.4% and a top-5 accuracy 80.4% on the validation set, using just the center crop.

- Download the Caffe CaffeNet binary file from http://dl.caffe.berkeleyvision.org/bvlc_reference_caffenet.caffemodel to <install_directory>/database/model/caffenet
- Download the Caffe CaffeNet depoly file from https://github.com/BVLC/caffe/blob/master/models/bvlc_reference_caffenet/deploy.prototxt to <install_directory>/src/database/model/caffenet
- Run the script ./databaseModelCaffeNetInterop.sh to perform the conversion

After the conversion, there should two files in <install_directory>/src/database/model/caffenet:

- databaseModelCaffeNet.m
- databaseModelCaffeNet.m.params

VGGNet [3]

VGGNet shows that a significant improvement on the prior-art configurations can be achieved by increasing the depth to 16-19 weight layers, which is substantially deeper than what has been used in the prior art. To reduce the number of parameters in such very deep networks, the researchers use very small 3×3 filters in all convolutional layers.

- Download the VGGNet binary file from http://www.robots.ox.ac.uk/~vgg/software/very_deep/caffe/VGG_ILSVRC_16_layers.caffemodel to <install_directory>/src/database/model/vggnet
- Download the VGGNet depoly file from https://gist.githubusercontent.com/ksimonyan/211839e770f7b VGG_ILSVRC_16_layers_deploy.prototxt to <install_directory>/src/database/model/vggnet
- Run the script ./databaseModelVGGNetInterop.sh to perform the conversion

After the conversion, there should two files in <install_directory>/src/database/model/caffenet:

- databaseModelVGGNet.m

- databaseModelVGGNet.m.params

MNIST A pretrained CNN for MNIST database is located at <install_directory>/src/database/model/mnist:

- databaseModelMNIST.m
- databaseModelMNIST.m.params

CIFAR10 A pretrained CNN for CIFAR10 database is located at <install_directory>/src/database/model/cifar10:

- databaseModelCIFAR10.m
- databaseModelCIFAR10.m.params

1.3 Convolutional Neural Networks (CNNs)

The TIML library provides a set of APIs that allow a user to implement a CNN architecture and perform training and testing. Common CNN layer types are supported in the current version and more will be added in future versions. Both image classification and scene labeling examples are provided.

This section briefly describes the use of CNN library APIs. For complete examples, refer to the folder `<install_directory>/src/app/cnn`

1.3.1 Training Parameters

The default training parameters structure is obtained by calling `timlCNNTrainingParamsDefault()`. Default parameters can be overridden by setting specific field values (refer to `timlCNNTrainingParams` for details).

1.3.2 Layers

This section describes the different CNN layers supported by the TIML library. To create a CNN, call `timlCNNCreateConvNeuralNetwork()`. Here is an example code block in `timlTestCNNSimpleTraining()` that generates a simple CNN with 8 layer types:

```
trainingParams              = timlCNNTrainingParamsDefault();
trainingParams.batchSize    = BATCH_SIZE;
trainingParams.learningRate = 0.1;
cnn = timlCNNCreateConvNeuralNetwork(trainingParams, 0);
inputParams       = timlCNNInputParamsDefault();
inputParams.scale = 1.0/256.0;
timlCNNAddInputLayer(cnn, IMAGE_ROW, IMAGE_COL, IMAGE_CHANNEL, inputParams);
       // input layer
timlCNNAddConvLayer(cnn, 5, 5, 1, 1, 6,
      timlCNNConvParamsDefault());                    // conv layer
timlCNNAddNonlinearLayer(cnn, Util_Relu);
           // relu layer
timlCNNAddPoolingLayer(cnn, 4, 4, 4, 4, CNN_MaxPooling,
      timlCNNPoolingParamsDefault()); // max pooling layer
timlCNNAddNormLayer(cnn, timlCNNNormParamsDefault());
                              // norm layer
timlCNNAddDropoutLayer(cnn, 0.5);
         // dropout layer
timlCNNAddLinearLayer(cnn, 10, timlCNNLinearParamsDefault())
      ;                           // linear layer
timlCNNAddNonlinearLayer(cnn, Util_Softmax);
           // softmax layer
timlCNNInitialize(cnn);
timlCNNReset(cnn);
```

1.3.2.1 Input

The input layer is responsible for preprocessing the raw input data. The user can choose to crop, scale, mirror, or subtract the mean form the raw image. The required parameters are the dimensions (row, column, channel) of the feature maps in the input layer. Refer to `timlCNNAddInputLayer()` for more details.

1.3.2.2 Convolutional

The convolutional layer performs 2D convolution or correlation between feature maps and kernels. The required parameters are the 2D kernel dimension (row, column), kernel strides along row and column and the output feature map channel. For each feature map in the previous layer, there will be a corresponding 2D kernel applied to it and links to each feature map in the next layer. Refer to `timlCNNAddConvLayer()` for more detail.

1.3.2.3 Nonlinear

The nonlinear layer implements 1 nonlinearity, supported types are:

- rectified linear unit

  $f(z) = \max(0, z)$

- sigmoid

  $f(z) = \frac{1}{1+e^{-z}}$

- softmax

  $f(z)_j = \frac{e^{-z_j}}{\sum_{i=0}^{K} e^{-z_i}}$

- tanh

  $f(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$

To add a nonlinear layer to the cnn structure, call the function `timlCNNAddNonlinearLayer()`. Note that for image classification applications, the last layer is most commonly chosen to be a softmax nonlinear layer that output an array of class probabilities that sum up to 1.

1.3.2.4 Pooling

The pooling layer performs a local maxing or averaging operation on the feature maps. The required parameters are the pooling kernel dimension (row, col), kernel strides along column and row and the pooling method. Refer to `timlCNNAddPoolingLayer()` for more details.

1.3.2.5 Normalization

The normalization layer performs local contrast normalization across channels:

$y^i = \frac{x^i}{(1+\frac{\alpha}{N}\sum_{j=\max(0,i-N/2)}^{\min(N-1,i+N/2)} x^j)^\beta}$,where x^i stands for the feature map in the i-th channel. The channel span N defaults to 5. α and β default to 0.001 and 0.75, respectively. Refer to `timlCNNAddNormLayer()` for more details.

1.3.2.6 Dropout

The dropout layer is used during training to help prevent overfitting. Each element in the feature map is set to 0 (deactivated) according to a preset probability in the training phase. Note the dropout layer is simply a pass through in the testing phase. Refer to `timlCNNAddDropoutLayer()` for more details.

1.3.2.7 Linear

The linear layer is also referred to as the inner product layer or fully connected layer. A traditional neural network layer, the feature map in the previous layer is first vectorized and then multiplied by a weight matrix to obtain the feature map of next layer. The required parameter is the dimension of the output feature map. Refer to `timlCNNAddLinearLayer()` for more details.

1.3.3 Memory

Once the setup of the cnn structure is complete, `timlCNNInitialize()` is called to allocate the memory. There are 3 levels of memory allocation specified inside `timlCNNTrainingParams`.

- Level 1 can be used both for training and testing and requires the most memory.
- Level 2 requires less memory but can only be used for testing.
- Level 3 uses even less memory by operating on a memory pool and can also only be used for testing.

Note that this function does not initialize the kernels or weights in the convolutional and linear layers. To perform that initialization the function `timlCNNReset()` is used.

The exact memory allocated in bytes can be obtained by calling `timlCNNMemory()`.

`timlCNNResize()` is used to resize the dimension of a CNN. Once the input layer dimension is changed, the dimension of the following layers will be re-calcualted and re-allocated.

`timlCNNClone()` creates a independent copy of a CNN. The function `timlCNNShareParams()` differs from `timlCNNClone()` in that the CNN structure returned by this function allocates its own feature map memory but points to the parameter memory of its target. Therefore, `timlCNNShareParams()` returns a structure that takes less memory compared with the structure returned by `timlCNNClone()`, which allocates its own parameter memory. The user should be careful when manipulating the shared copy of a CNN as it may write to the parameters of its target. `timlCNNShareParams()` is used primarily to accelerate the training and testing of a CNN. Multiple shared copies of a CNN can work in parallel by using OpenMP.

To free the space allocated by the CNN, call the function `timlCNNDelete()`.

1.3.4 Utility Functions

Utility functions aim to perform miscellaneous functions that related to IO.

The phase of a CNN can be set to training or testing by calling `timlCNNSetMode()`. A CNN allocated with level 2 or 3 is not allowed to be set to the training mode as no memory is allocated to run the back propagation algorithm. In the testing mode, only forward propagation will be performed.

CNN structure information can be printed to the console by using `timlCNNPrint()`. `timlCNNGetLayerNum()` returns the number of layers of the CNN. `timlCNNGetParamsNum()` returns the total number of parameters in the CNN.

`timlCNNWriteToFile()` writes the cnn structure into a combination of text and binary files. The key parameter of this function is the `timlUtilParamsLevel`, which specifies the level of details of the writing.

- Level 1 only writes the network structure to a text file without specifying the parameters or feature maps of the cnn. The text file is formatted in a syntax that is compatible with Matlab script.
- Level 2 writes both the network structure text file and the binary parameter file.
- Level 3 writes one more state binary files. Level 3 is only used for debugging purpose. Note that the path of the binary files to read is written into the text file.

Similarly, `timlCNNReadFromFile()` can be used to read a CNN from text or binary files.

One example of the generated text file is shown below:

```
paramsBinaryFileName = ’./database/test/cnn/timl_cnn_simple_config.m.params’;
stateBinaryFileName  = ’./database/test/cnn/timl_cnn_simple_config.m.state’;

cnn.params.count          = 0;
cnn.params.batchSize      = 100;
cnn.params.epoch          = 1;
cnn.params.learningRate   = 0.1000;
cnn.params.momentum       = 0.0000;
cnn.params.phase          = 0;
cnn.params.allocatorLevel = 0;
cnn.params.costType       = 0;

layerNum                                    = 8;
cnn.layer(1).id                             = 1;
cnn.layer(1).type                           = 0;
cnn.layer(1).row                            = 28;
cnn.layer(1).col                            = 28;
cnn.layer(1).channel                        = 1;
cnn.layer(1).inputParams.row                = 28;
cnn.layer(1).inputParams.col                = 28;
cnn.layer(1).inputParams.channel            = 1;
cnn.layer(1).inputParams.scale              = 1.0000;
cnn.layer(1).inputParams.trainingCropType   = 0;
cnn.layer(1).inputParams.trainingMirrorType = 1;
cnn.layer(1).inputParams.testingCropType    = 0;
cnn.layer(1).inputParams.testingMirrorType  = 1;

cnn.layer(2).id                                 = 2;
cnn.layer(2).type                               = 1;
cnn.layer(2).row                                = 24;
cnn.layer(2).col                                = 24;
cnn.layer(2).channel                            = 6;
cnn.layer(2).convParams.kernelRow               = 5;
cnn.layer(2).convParams.kernelCol               = 5;
cnn.layer(2).convParams.padUp                   = 0;
cnn.layer(2).convParams.padDown                 = 0;
cnn.layer(2).convParams.padLeft                 = 0;
cnn.layer(2).convParams.padRight                = 0;
cnn.layer(2).convParams.strideX                 = 1;
cnn.layer(2).convParams.strideY                 = 1;
cnn.layer(2).convParams.inputFeatureMapChannel  = 1;
cnn.layer(2).convParams.outputFeatureMapChannel = 6;
cnn.layer(2).convParams.type                    = 0;
cnn.layer(2).convParams.kernelDecayFactor       = 1.0000;
cnn.layer(2).convParams.kernelInit.type         = 3;
cnn.layer(2).convParams.kernelLearningFactor    = 1.0000;
cnn.layer(2).convParams.biasInit.type           = 0;
cnn.layer(2).convParams.biasLearningFactor      = 1.0000;

cnn.layer(3).id                   = 3;
cnn.layer(3).type                 = 3;
cnn.layer(3).row                  = 24;
cnn.layer(3).col                  = 24;
cnn.layer(3).channel              = 6;
cnn.layer(3).nonlinearParams.type = 0;

cnn.layer(4).id                     = 4;
cnn.layer(4).type                   = 2;
cnn.layer(4).row                    = 12;
cnn.layer(4).col                    = 12;
cnn.layer(4).channel                = 6;
cnn.layer(4).poolingParams.type     = 0;
cnn.layer(4).poolingParams.scaleRow = 2;
cnn.layer(4).poolingParams.scaleCol = 2;
cnn.layer(4).poolingParams.padUp    = 0;
cnn.layer(4).poolingParams.padDown  = 0;
cnn.layer(4).poolingParams.padLeft  = 0;
cnn.layer(4).poolingParams.padRight = 0;
cnn.layer(4).poolingParams.strideX  = 2;
cnn.layer(4).poolingParams.strideY  = 2;

cnn.layer(5).id                 = 5;
cnn.layer(5).type               = 6;
cnn.layer(5).row                = 12;
cnn.layer(5).col                = 12;
cnn.layer(5).channel            = 6;
cnn.layer(5).dropoutParams.prob = 0.5000

cnn.layer(6).id               = 6;
cnn.layer(6).type             = 5;
cnn.layer(6).row              = 12;
cnn.layer(6).col              = 12;
cnn.layer(6).channel          = 6;
cnn.layer(6).normParams.type  = 0;
cnn.layer(6).normParams.N     = 5;
cnn.layer(6).normParams.alpha = 0.0010;
cnn.layer(6).normParams.beta  = 0.7500;
cnn.layer(7).id                             = 7;
cnn.layer(7).type                           = 4;
cnn.layer(7).row                            = 1;
cnn.layer(7).col                            = 1;
cnn.layer(7).channel                        = 10;
cnn.layer(7).linearParams.dim               = 10;
cnn.layer(7).linearParams.prevDim           = 864;
cnn.layer(7).linearParams.weightDecayFactor    = 1.0000;
cnn.layer(7).linearParams.weightInit.type      = 3;
cnn.layer(7).linearParams.weightLearningFactor = 1.0000;
cnn.layer(7).linearParams.biasInit.type        = 0;
cnn.layer(7).linearParams.biasLearningFactor   = 1.0000;

cnn.layer(8).id                   = 8;
cnn.layer(8).type                 = 3;
cnn.layer(8).row                  = 1;
cnn.layer(8).col                  = 1;
cnn.layer(8).channel              = 10;
cnn.layer(8).nonlinearParams.type = 1;
```

Note that the text file is formatted using Matlab syntax such that the text file can be used as a script to obtain a cnn structure object.

1.3.5 Training

The following code block in `testCNNSimpleTraining()` performs training:

```
// read MNIST database
 printf("2. Read the MNIST database\n");
 timlUtilReadMNIST(DATABASE_PATH, &training, &testing);

 // training
 printf("3. Start training\n");
 clock_gettime(CLOCK_REALTIME, &startTime);
 for (i = 0; i < batchNum; i++) {
    timlCNNSupervisedTrainingWithLabelBatchMode(cnn, training.
      data + i*batchSize*dim, training.label + i*batchSize, dim, batchSize);
 }
 clock_gettime(CLOCK_REALTIME, &endTime);
 trainingTime = timlUtilDiffTime(startTime, endTime);
 printf("Training time = %.2f s.\n", trainingTime/1000000.0);
```

In this example, the training and testing image data set structures are read from the MNIST database and then passed into `timlCNNSupervisedTrainingWithLabelBatchMode()`. Note that there is also a multi-thread version of this function that called `timlCNNSupervisedTrainingWithLabelBatchMode()`. The training time is returned by `timlUtilDiffTime()` in microsecond precision.

1.3.6 Testing

The following code block in `testCNNSimpleTesting()` performs testing:

```
for(i = 0; i < testing.num; i++)
{
    label = timlCNNClassifyTop1SingleMode(cnn, testing.data + i*dim, dim);
    if (label != testing.label[i]) misClassifyNum++;
}
```

`timlCNNClassifyTop1SingleMode()` returns the top 1 label generated by the CNN. A similar function `timlCNNClassifyTopNBatchMode()` that returns the top N labels together with their corresponding probabilities for a batch of data. Note that there is also a multi-thread version of this function called `timlCNNClassifyTopNBatchModeOpenMP()` that classifies a batch of images using OpenMP.

1.4 Applications

Application source code is located at `<install_directory>/src/app`. To run the applications, change the directory to the corresponding binary files and run the applications.

There are 1 application example for Caffe to TIML CNN model interoperation:

- appCNNInteropCaffe: Caffe interoperation

There are 2 application example for Caffe to TIML CNN database conversion:

- appCNNConvertImageNet: ImagNet conversion
- appCNNConvertSBD: SBD conversion

There are 8 application examples for CNN classification:

- appCNNClassMNISTTraining(): MNIST database training
- appCNNClassMNISTTesting(): MNIST database testing
- appCNNClassCIFAR10Training(): CIFAR10 database training
- appCNNClassCIFAR10Testing(): CIFAR10 database testing
- appCNNClassCaffeNetTraining(): CaffeNet database training
- appCNNClassCaffeNetTesting(): CaffeNet database testing
- appCNNClassAlexNetTesting(): AlexNet database training
- appCNNClassVGGNetTesting(): VGGNet database testing

There are 2 application examples for CNN scene labeling:

- appCNNSceneSBDTraining(): scene labeling database training
- appCNNSceneSBDTesting(): scene labeling database testing

1.4.1 Classification

Training and testing a CNN for image classification is very straightforward to implement using the TIML library. Let's take the MNIST database for example. Here is one code block in appCNNClassMNISTTraining(). There are 3 major steps. First, build up the CNN structure using the library API. Next, read the database. Third, apply the training function on the database. After the training, you can write the trained network to file(s) using timlCNNWriteToFile().

```
// setup CNN
printf("1. Build up the CNN\n");
timlConvNeuralNetwork *cnn = timlCNNCreateConvNeuralNetwork
      (timlCNNTrainingParamsDefault(), 0);
cnn->params.learningRate = LEARN_RATE;
cnn->params.batchSize    = BATCH_SIZE;
inputParams = timlCNNInputParamsDefault();
inputParams.scale = 1.0/256.0;
timlCNNAddInputLayer(cnn, IMAGE_ROW, IMAGE_COL, IMAGE_CHANNEL, inputParams);
        // input layer
convParams = timlCNNConvParamsDefault();
convParams.kernelInit.type = Util_Xavier;
timlCNNAddConvLayer(cnn, 5, 5, 1, 1, 20, convParams);
       // conv layer
poolingParams = timlCNNPoolingParamsDefault();
timlCNNAddPoolingLayer(cnn, 2, 2, 2, 2, CNN_MaxPooling, poolingParams);
          // max pooling layer
convParams = timlCNNConvParamsDefault();
convParams.kernelInit.type = Util_Xavier;
timlCNNAddConvLayer(cnn, 5, 5, 1, 1, 50, convParams);
       // conv layer
timlCNNAddPoolingLayer(cnn, 2, 2, 2, 2, CNN_MaxPooling,
      timlCNNPoolingParamsDefault());  // max pooling layer
timlCNNAddLinearLayer(cnn, 500, timlCNNLinearParamsDefault()
      );                           // linear layer
timlCNNAddNonlinearLayer(cnn, Util_Relu);
            // relu layer
```

```
timlCNNAddLinearLayer(cnn, 10, timlCNNLinearParamsDefault())
      ;                             // linear layer
timlCNNAddNonlinearLayer(cnn, Util_Softmax);
            // softmax layer
timlCNNInitialize(cnn);
timlCNNReset(cnn);
mem = timlCNNMemory(cnn);
timlCNNPrint(cnn);
printf("CNN memory allocation = %.10f MB.\n", (float) mem/1024.0/1024.0);
printf("CNN parameter #       = %ld.\n", timlCNNGetParamsNum(cnn));

// read MNIST database
printf("2. Read the MNIST database\n");
timlUtilReadMNIST(DATABASE_PATH, &training, &testing);

// training
printf("3. Start training\n");
clock_gettime(CLOCK_REALTIME, &startTime);
for (i = 0; i < batchNum; i++) {
   timlCNNSupervisedTrainingWithLabelBatchMode(cnn, training.
      data + i*batchSize*dim, training.label + i*batchSize, dim, batchSize);
}
clock_gettime(CLOCK_REALTIME, &endTime);
trainingTime = timlUtilDiffTime(startTime, endTime);
printf("Training time = %.2f s.\n", trainingTime/1000000.0);
```

Deploying or testing the CNN is even simpler. Here is one code block in appCNNClassMNISTTesting(). Again, there are 3 major steps. First, read the CNN structure from file(s). Next, read the testing database. Third, apply the testing function to produced the labels generated by the CNN.

```
// read CNN config
printf("1. Read CNN config\n");
timlConvNeuralNetwork *cnn = timlCNNReadFromFile(MODEL_PATH, 0);
timlCNNSetMode(cnn, Util_Tes@subsection subsectionSceneLabeling Scene Labelingt);
mem = timlCNNMemory(cnn);
timlCNNPrint(cnn);
printf("CNN memory allocation = %.10f MB.\n", (float)mem/1024.0/1024.0);
printf("CNN parameter #       = %lu.\n", timlCNNGetParamsNum(cnn));

// read MNIST database
printf("2. Read MNIST database\n");
timlUtilReadMNIST(DATABASE_PATH, &training, &testing);

// testing
printf("3. Start testing\n");
clock_gettime(CLOCK_REALTIME, &startTime);
timlCNNClassifyTopNBatchMode(cnn, testing.data, dim, testing.num, label, NULL,
      topN);
clock_gettime(CLOCK_REALTIME, &endTime);
testingTime = timlUtilDiffTime(startTime, endTime);
classifyNum = timlUtilClassifyAccuracy(label, topN, testing.num, testing.label);
classifyPercent = (float)classifyNum/(float)testing.num;
printf("Testing time      = %.3f s\n", testingTime/1000000.0);
printf("Classify accuracy = %.3f %%\n", classifyPercent*100.00);
```

1.4.2 Scene Labeling

Scene labeling consists in labeling each pixel in an image with the category of the object it belongs to. Instread of producing one lable for the entire image, scene labeling produces one label for each pixel in the image. Therefore, training a CNN for scene labeling is slightly different from training a CNN for classification purpose. The setup of the CNN follows the same procedure:

```
// build up the CNN
printf("1. Build up the CNN\n");
cnn = timlCNNCreateConvNeuralNetwork(
      timlCNNTrainingParamsDefault(), 0);
timlCNNAddInputLayer(cnn, PATCH_SIZE, PATCH_SIZE, IMAGE_CHANNEL,
      timlCNNInputParamsDefault());
timlCNNAddConvLayer(cnn, 6, 6, 1, 1, 25,
      timlCNNConvParamsDefault());                  // conv layer
timlCNNAddNonlinearLayer(cnn, Util_Tanh);
            // tanh layer
timlCNNAddPoolingLayer(cnn, 8, 8, 8, 8, CNN_MaxPooling,
      timlCNNPoolingParamsDefault());  // max pooling layer
timlCNNAddConvLayer(cnn, 3, 3, 1, 1, 50,
      timlCNNConvParamsDefault());                  // conv layer
timlCNNAddNonlinearLayer(cnn, Util_Tanh);
            // tanh layer
```

```
timlCNNAddPoolingLayer(cnn, 2, 2, 2, 2, CNN_MaxPooling,
      timlCNNPoolingParamsDefault());  // max pooling layer
timlCNNAddLinearLayer(cnn, 8, timlCNNLinearParamsDefault());
                                   // linear layer
timlCNNAddNonlinearLayer(cnn, Util_Softmax);
            // softmax layer
```

The second step is to setup the training database:

```
slTraining.num               = TRAIN_IMAGE_NUM;
slTraining.row               = IMAGE_ROW;
slTraining.col               = IMAGE_COL;
slTraining.channel           = IMAGE_CHANNEL;
slTraining.patchSize         = PATCH_SIZE;
slTraining.imageFileNameStr = TRAIN_IMAGE_PATH;
slTraining.labelFileNameStr = TRAIN_LABEL_PATH;
```

In this example, 450 images are used for training and 143 images are used for testing. The dimension of the images are 240∗320∗3. There are 450 text file labels in the training database, each with a 240∗320 label matrix. The label integer ranges from -1 to 7 (8 classes) with -1 indicating an unlabeled pixel. The image and label text file name must follow the format as indicated in the imageFileNameStr and labelFileNameStr. For each pixel in the image, a 133∗133 patch centered at the pixel is passed into the CNN to output a label.

To train the database, the CNN first randomly selects an image and then randomly selects a pixel. Next, a patch centered at that pixel is passed to the CNN for training. The pixels are randomly selected without replacement, with 240∗320∗450 iterations needed to sweep through the entire database 1x. There is a specialized training function:

```
appCNNSceneSupervisedTraining(cnn, &slTraining);
```

To test one image, a natural way is to generate a patch for each pixel which requires 240∗320 forward passes through the CNN. However, there is more efficient way to do this which exploits the convolutional structure. First, pad zeros on the test image and shift the image in all four directions. The number of shifted and zero padded images is proportional to the number of pooling layers in the CNN. Next, pass the shifted and zero padded image to the trained CNN. Finally, merge the output feature maps into a label matrix for the image. Using this approach, the input feature map size is no longer the patch size which is 133∗133 in the above example. We need to re-calculate the input feature map dimensions and use function `timlCNNResize()` to resize the CNN. Note that the above cnn has gone through 2 max pooling layers which scales down the feature map by a factor of 2∗8 in the row and column dimensions. The formula to calculate the resized row and col for the input layer is:

```
printf("3. Resize the feature maps\n");
resolutionLossRow = 8*2; // resolution loss due to max pooling
resolutionLossCol = 8*2; // resolution loss due to max pooling
resizeRow = slTraining.row + (slTraining.patchSize/2)*2 - (resolutionLossRow - 1);
resizeCol = slTraining.col + (slTraining.patchSize/2)*2 - (resolutionLossCol - 1);
timlCNNResize(cnn, resizeRow, resizeCol, slTraining.channel);
```

Multiple forward passes through the same CNN can be performed in parallel. First, create multilple shared copies of the CNN to form a team called cnnTeam. Then we can call the function `appCNNSceneClassifyOpenMP()` to perform the labeling operation. A single-thread version of the function also exists as `appCNNSceneClassify()`; The parameter scale in `appCNNSceneClassifyOpenMP()` stands for the scale down factor of the generated label matrix. When scale == 1, no downscaling is performed. When scale == 4, the generated label matrix would be of size 60∗80 and then upscaled to 240∗320 to match the size of the image. Downscaling can effectively reduce the labeling time.

```
// create cnnTeam
cnnTeam[0] = cnn;
for (i = 1; i < thread; i++) {
   cnnTeam[i] = timlCNNShareParams(cnn, 0);
}

// testing
printf("2. Start testing\n");
for (i = 0; i < slTesting.num; i++) {
   printf("Read image %03d.jpg\n", i);
   sprintf(str, slTesting.imageFileNameStr, i);
   timlUtilReadFixedSizeJPEG(str, image, slTesting.row, slTesting.col, slTesting.
      channel);
   clock_gettime(CLOCK_REALTIME, &startTime);
   appCNNSceneClassifyOpenMP(cnnTeam, thread, image, slTesting.row, slTesting.col,
       slTesting.channel, labelMatrix, scale);
   clock_gettime(CLOCK_REALTIME, &endTime);
   testingTime = timlUtilDiffTime(startTime, endTime);

   // read true label
   sprintf(str, slTesting.labelFileNameStr, i);
   fp = fopen(str, "rt");
   for (n = 0; n < slTesting.row; n++) {
      for (m = 0; m < slTesting.col; m++) {
         fscanf(fp, "%d", trueLabelMatrix + n*slTesting.col + m);
      }
      fscanf(fp, "\n");
   }
   fclose(fp);

   // calculate accuracy
   labelAccuracy = appCNNSceneAccuracy(labelMatrix, trueLabelMatrix, slTesting.row*
      slTesting.col);
   printf("Test image %03d label accuracy = %.2f %%\n", i, 100.0*labelAccuracy);
   printf("Test image %03d time           = %.3f s\n", i, testingTime/1000000.0);
}
```

1.5 Benchmarks

The benchmark source code is located at `<install_directory>/src/benchmark`.

There are 2 examples for CNN classification benchmark:

- `benchmarkCNNClassCaffeNetTesting()` : CaffeNet benchmark
- `benchmarkCNNClassVGGNetTesting()` : VGGNet benchmark

These two functions will benchmark the memory usage and processing time for each individual layers in the network.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

cnn . . . 41
util . . . 78
app . . . 103
- appCNN . . . 104
  - appCNNClass . . . 25
  - appCNNConvertImageNet . . . 26
  - appCNNConvertSBD . . . 27
  - appCNNInteropCaffe . . . 28
  - appCNNScene . . . 35

benchmark . . . 105
- benchmarkCNN . . . 106
  - benchmarkCNNClass . . . 40

test . . . 107
- testCNN . . . 100
- testUtil . . . 102

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

timlCNNLayer 109
timlConvNeuralNetwork 110
appCNNSceneDataSet 110
timlCNNConvParams 111
timlCNNDataSet 112
timlCNNDropoutParams 112
timlCNNInputParams 113
timlCNNLinearParams 114
timlCNNNonlinearParams 114
timlCNNNormParams 115
timlCNNPoolingParams 115
timlCNNTrainingParams 116
timlUtilImage 117
timlUtilImageSet 117
timlUtilInitializer 117

Chapter 4

File Index

4.1 File List

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

appCNNClass.h . . . 119
appCNNClassCIFAR10Testing.c . . . 119
appCNNClassCIFAR10Training.c . . . 120
appCNNClassImageNetAlexNetTesting.c . . . 120
appCNNClassImageNetCaffeNetTesting.c . . . 121
appCNNClassImageNetCaffeNetTraining.c . . . 121
appCNNClassImageNetVGGNetTesting.c . . . 121
appCNNClassMNISTTesting.c . . . 122
appCNNClassMNISTTraining.c . . . 122
appCNNConvertImageNet.cpp . . . 123
appCNNConvertImageNet.hpp . . . 123
appCNNConvertSBD.cpp . . . 124
appCNNConvertSBD.hpp . . . 124
appCNNConvertSBDShuffle.cpp . . . 124
appCNNInteropCaffe.cpp . . . 125
appCNNInteropCaffe.hpp . . . 125
appCNNInteropCaffeConvert.cpp . . . 126
appCNNInteropCaffeConvLayerConvert.cpp . . . 126
appCNNInteropCaffeConvLayerPermuteKernel.cpp . . . 127
appCNNInteropCaffeDropoutLayerConvert.cpp . . . 127
appCNNInteropCaffeFillBlockDiagonalMatrix.cpp . . . 127
appCNNInteropCaffeFlipKernelMatrix.cpp . . . 127
appCNNInteropCaffeFlipMatrixFloat.cpp . . . 128
appCNNInteropCaffeLayerTypeConvert.cpp . . . 128
appCNNInteropCaffeLinearLayerConvert.cpp . . . 128
appCNNInteropCaffeNonlinearLayerConvert.cpp . . . 128
appCNNInteropCaffeNonlinearTypeConvert.cpp . . . 128
appCNNInteropCaffeNormLayerConvert.cpp . . . 129
appCNNInteropCaffePermuteMean.cpp . . . 129
appCNNInteropCaffePoolingLayerConvert.cpp . . . 129
appCNNInteropCaffePoolingTypeConvert.cpp . . . 129
appCNNInteropCaffeReadMean.cpp . . . 130
appCNNInteropCaffeReadProtoFromBinaryFile.cpp . . . 130
appCNNInteropCaffeReadProtoFromTextFile.cpp . . . 130
appCNNScene.h . . . 130
appCNNSceneAccuracy.c . . . 131
appCNNSceneClassify.c . . . 131
appCNNSceneClassifyOpenMP.c . . . 131

appCNNSceneGetLabel.c . . . 132
appCNNSceneGetPatch.c . . . 132
appCNNSceneLabelMatrix.c . . . 132
appCNNSceneSBDTesting.c . . . 132
appCNNSceneSBDTraining.c . . . 133
appCNNSceneShuffleIdx.c . . . 133
appCNNSceneSupervisedTraining.c . . . 133
benchmarkCNNClass.h . . . 134
benchmarkCNNClassCaffeNetTesting.c . . . 134
benchmarkCNNClassVGGNetTesting.c . . . 134
testCNN.c . . . 135
testCNN.h . . . 135
testCNNSimpleClone.c . . . 135
testCNNSimpleIO.c . . . 136
testCNNSimpleProfile.c . . . 136
testCNNSimpleResize.c . . . 137
testCNNSimpleShare.c . . . 137
testCNNSimpleTraining.c . . . 138
testUtil.c . . . 138
testUtil.h . . . 138
testUtilBLAS.c . . . 138
testUtilConv2.c . . . 139
testUtilSort.c . . . 139
timl.h
Timl public APIs . . . 139
timlCNN.h . . . 141
timlCNNAddConvLayer.c . . . 145
timlCNNAddDropoutLayer.c . . . 145
timlCNNAddInputLayer.c . . . 145
timlCNNAddLinearLayer.c . . . 145
timlCNNAddNonlinearLayer.c . . . 146
timlCNNAddNormLayer.c . . . 146
timlCNNAddPoolingLayer.c . . . 146
timlCNNAssignDevice.c . . . 146
timlCNNBackPropagation.c . . . 147
timlCNNClassifyTop1SingleMode.c . . . 147
timlCNNClassifyTopNBatchMode.c . . . 147
timlCNNClassifyTopNBatchModeOpenMP.c . . . 147
timlCNNClassifyTopNTeamModeOpenMP.c . . . 147
timlCNNClone.c . . . 148
timlCNNConvBackPropagation.c . . . 148
timlCNNConvForwardPropagation.c . . . 148
timlCNNConvInitialize.c . . . 148
timlCNNConvParamsDefault.c . . . 149
timlCNNConvReadFromBinaryFile.c . . . 149
timlCNNConvReadFromTextFile.c . . . 149
timlCNNConvShareParams.c . . . 149
timlCNNConvUpdateParams.c . . . 149
timlCNNConvWriteToFile.c . . . 150
timlCNNCostWithLabel.c . . . 150
timlCNNCreateConvNeuralNetwork.c . . . 150
timlCNNDelete.c . . . 150
timlCNNDeleteConvLayer.c . . . 151
timlCNNDeleteDropoutLayer.c . . . 151
timlCNNDeleteInputLayer.c . . . 151
timlCNNDeleteLinearLayer.c . . . 151
timlCNNDeleteNonlinearLayer.c . . . 151
timlCNNDeleteNormLayer.c . . . 152

timlCNNDeletePoolingLayer.c . . . 152
timlCNNDropoutBackPropagation.c . . . 152
timlCNNDropoutForwardPropagation.c . . . 152
timlCNNDropoutInitialize.c . . . 153
timlCNNDropoutReadFromTextFile.c . . . 153
timlCNNDropoutWriteToFile.c . . . 153
timlCNNForwardPropagation.c . . . 153
timlCNNGetLayerNum.c . . . 153
timlCNNGetParamsNum.c . . . 154
timlCNNInitialize.c . . . 154
timlCNNInputInitialize.c . . . 154
timlCNNInputParamsDefault.c . . . 154
timlCNNInputReadFromBinaryFile.c . . . 155
timlCNNInputReadFromTextFile.c . . . 155
timlCNNInputShareParams.c . . . 155
timlCNNInputWriteToFile.c . . . 155
timlCNNLayerTypeStr.c . . . 155
timlCNNLinearBackPropagation.c . . . 156
timlCNNLinearForwardPropagation.c . . . 156
timlCNNLinearInitialize.c . . . 156
timlCNNLinearParamsDefault.c . . . 156
timlCNNLinearReadFromBinaryFile.c . . . 157
timlCNNLinearReadFromTextFile.c . . . 157
timlCNNLinearShareParams.c . . . 157
timlCNNLinearUpdateParams.c . . . 157
timlCNNLinearWriteToFile.c . . . 157
timlCNNMaxPoolingBackPropagation.c . . . 158
timlCNNMaxPoolingForwardPropagation.c . . . 158
timlCNNMeanPoolingBackPropagation.c . . . 158
timlCNNMeanPoolingForwardPropagation.c . . . 158
timlCNNMemory.c . . . 159
timlCNNMemPoolSize.c . . . 159
timlCNNNonlinearBackPropagation.c . . . 159
timlCNNNonlinearForwardPropagation.c . . . 159
timlCNNNonlinearInitialize.c . . . 159
timlCNNNonlinearParamsDefault.c . . . 160
timlCNNNonlinearReadFromTextFile.c . . . 160
timlCNNNonlinearWriteToFile.c . . . 160
timlCNNNormBackPropagation.c . . . 160
timlCNNNormForwardPropagation.c . . . 161
timlCNNNormInitialize.c . . . 161
timlCNNNormParamsDefault.c . . . 161
timlCNNNormWriteToFile.c . . . 161
timlCNNPoolingBackPropagation.c . . . 161
timlCNNPoolingForwardPropagation.c . . . 162
timlCNNPoolingInitialize.c . . . 162
timlCNNPoolingParamsDefault.c . . . 162
timlCNNPoolingReadFromTextFile.c . . . 162
timlCNNPoolingWriteToFile.c . . . 163
timlCNNPrint.c . . . 163
timlCNNProfile.c . . . 163
timlCNNReadFromFile.c . . . 163
timlCNNReset.c . . . 163
timlCNNResetConvLayer.c . . . 164
timlCNNResetDropoutLayer.c . . . 164
timlCNNResetInputLayer.c . . . 164
timlCNNResetLinearLayer.c . . . 164
timlCNNResetNonlinearLayer.c . . . 165

timlCNNResetNormLayer.c . . . 165
timlCNNResetPoolingLayer.c . . . 165
timlCNNResize.c . . . 165
timlCNNSetMode.c . . . 165
timlCNNShareParams.c . . . 166
timlCNNSupervisedTrainingWithLabelBatchMode.c . . . 166
timlCNNSupervisedTrainingWithLabelBatchModeOpenMP.c . . . 166
timlCNNTrainingParamsDefault.c . . . 166
timlCNNTrainingParamsReadFromTextFile.c . . . 167
timlCNNTrainingParamsWriteToFile.c . . . 167
timlCNNUpdateParams.c . . . 167
timlCNNWriteToFile.c . . . 167
timlUtil.h . . . 168
timlUtilBLAS.c . . . 172
timlUtilClassifyAccuracy.c . . . 172
timlUtilConv2Full.c . . . 173
timlUtilConv2ImageReshape.c . . . 173
timlUtilConv2ImageReshapeBack.c . . . 173
timlUtilConv2ImageReshapeIndex.c . . . 173
timlUtilConv2Valid.c . . . 173
timlUtilCorr2Full.c . . . 174
timlUtilDiffTime.c . . . 174
timlUtilElementWiseFunction.c . . . 174
timlUtilElementWiseMultiply.c . . . 174
timlUtilFread.c . . . 175
timlUtilFree.c . . . 175
timlUtilFwrite.c . . . 175
timlUtilLocalContrastNormalize.c . . . 175
timlUtilLocalContrastUnnormalize.c . . . 175
timlUtilMalloc.c . . . 176
timlUtilMasking.c . . . 176
timlUtilMaxPooling.c . . . 176
timlUtilMeanPooling.c . . . 176
timlUtilMeanSquareError.c . . . 177
timlUtilMultinomialCrossEntropy.c . . . 177
timlUtilRandContinuousUniformRNG.c . . . 177
timlUtilRandDiscreteUniformRNG.c . . . 177
timlUtilRandNormalRNG.c . . . 178
timlUtilRandPerm.c . . . 178
timlUtilReadCIFAR10.c . . . 178
timlUtilReadFixedSizeJPEG.c . . . 178
timlUtilReadJPEG.c . . . 179
timlUtilReadMNIST.c . . . 179
timlUtilRelu.c . . . 179
timlUtilReluDerivative.c . . . 179
timlUtilReverseEndian32.c . . . 180
timlUtilScanJPEG.c . . . 180
timlUtilSigmoid.c . . . 180
timlUtilSigmoidDerivative.c . . . 180
timlUtilSoftmax.c . . . 180
timlUtilSubtract.c . . . 181
timlUtilTanh.c . . . 181
timlUtilTanhDerivative.c . . . 181
timlUtilTransform.c . . . 181
timlUtilUndoMaxPooling.c . . . 182
timlUtilUndoMeanPooling.c . . . 182
timlUtilUnmasking.c . . . 182
timlUtilVectorMaxFloat.c . . . 182

timlUtilVectorMaxIndexFloat.c 182
timlUtilVectorResetFloat.c 183
timlUtilVectorResetInt.c 183
timlUtilVectorSortFloat.c 183
timlUtilVectorSortIndexFloat.c 183
timlUtilVectorSumFloat.c 184

Chapter 5

Module Documentation

5.1 appCNNClass

CNN classification application.

Functions

- int appCNNClassMNISTTraining ()

  MNIST training example.
- int appCNNClassMNISTTesting ()

  MNIST classification testing example.
- int appCNNClassCIFAR10Training ()

  CIFAR10 training example.
- int appCNNClassCIFAR10Testing ()

  CIFAR10 testing example.
- int appCNNClassImageNetCaffeNetTesting ()

  CaffeNet classification testing example.
- int appCNNClassImageNetCaffeNetTraining ()

  CaffeNet training example.
- int appCNNClassImageNetAlexNetTesting ()

  AlexNet classification testing example.
- int appCNNClassImageNetVGGNetTesting ()

  VGGNet classification testing example.

5.1.1 Detailed Description

CNN classification application.

5.2 appCNNConvertImageNet

ImageNet 2012 database conversion applications.

Functions

- int main (int argc, char ∗argv[])

  Convert ImageNet database to have uniform size 256∗256.
- int appCNNConvertImageNetShuffle (char ∗∗names, int ∗labels, int n)

  Shuffle the images.

5.2.1 Detailed Description

ImageNet 2012 database conversion applications.

5.2.2 Function Documentation

5.2.2.1 int appCNNConvertImageNetShuffle (char ∗∗ *names,* int ∗ *labels,* int *n*)

Shuffle the images.

Parameters

in	*names*	Image name array
in	*labels*	Image label array
in	*n*	Array size

Returns

Error code

5.2.2.2 int main (int *argc,* char ∗ *argv[]*)

Convert ImageNet database to have uniform size 256∗256.

argv[0] = program name argv[1] = original training image folder argv[2] = original training label file argv[3] = converted training image folder argv[4] = original testing image folder argv[5] = original testing label file argv[6] = converted testing image folder argv[7] = training image number (optional) argv[8] = testing image number (optional)

Returns

Error code

5.3 appCNNConvertSBD

Stanford background dataset conversion applications.

Functions

- int main (int argc, char ∗argv[ ])

  Convert Stanford Background database to have uniform size 240∗320.
- int appCNNConvertSBDShuffle (char ∗∗names, int n)

  Shuffle the images.

5.3.1 Detailed Description

Stanford background dataset conversion applications.

5.3.2 Function Documentation

5.3.2.1 int appCNNConvertSBDShuffle (char ∗∗ *names,* int *n*)

Shuffle the images.

Parameters

in	*names*	Image name array
in	*n*	Array size

Returns

Error code

5.3.2.2 int main (int *argc,* char ∗ *argv[ ]*)

Convert Stanford Background database to have uniform size 240∗320.

argv[0] = program name argv[1] = original image folder argv[2] = original label file argv[3] = original image index file argv[4] = converted training folder argv[5] = converted testing folder argv[6] = training image number (optional) argv[7] = testing image number (optional)

Returns

Error code

5.4 appCNNInteropCaffe

CNN Caffe interoperation applications.

Functions

- int main (int argc, char ∗argv[ ])

  Caffe to TIML CNN model converter.
- bool appCNNInteropCaffeReadProtoFromTextFile (const char ∗fileName, Message ∗proto)

  Caffe read proto from text file.
- bool appCNNInteropCaffeReadProtoFromBinaryFile (const char ∗fileName, Message ∗proto)

  Caffe read proto from binary file.
- int appCNNInteropCaffeFlipMatrixFloat (float ∗a, int m, int n)

  Flip a matrix.
- int appCNNInteropCaffeFlipKernelMatrix (float ∗kernel, int kernelRow, int kernelCol, int inputChannel, int outputChannel)

  Flip the kernels.
- int appCNNInteropCaffeFillBlockDiagonalMatrix (float ∗a, int M, int N, int group, float ∗b)

  Fill a block diagonal matrix.
- timlUtilActivationType appCNNInteropCaffeNonlinearTypeConvert (LayerParameter_LayerType type)

  Caffe nonlinear layer type conversion.
- timlCNNLayerType appCNNInteropCaffeLayerTypeConvert (LayerParameter_LayerType type)

  Caffe to TIML CNN layer type conversion.
- timlCNNPoolingType appCNNInteropCaffePoolingTypeConvert (PoolingParameter_PoolMethod method)

  Caffe pooling type conversion.
- timlConvNeuralNetwork ∗ appCNNInteropCaffeConvert (const char ∗netStructurePrototxtFileName, const char ∗netParamPrototxtFileName)

  Convert Caffe to TIML CNN.
- int appCNNInteropCaffeConvLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Convert Caffe conv layer.
- int appCNNInteropCaffeConvLayerPermuteKernel (timlCNNLayer ∗layer)

  Change the kernel from BGR squence to RGB.
- int appCNNInteropCaffePoolingLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe pooling layer conversion.
- int appCNNInteropCaffeNormLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe norm layer conversion.
- int appCNNInteropCaffeLinearLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe linear layer conversion.
- int appCNNInteropCaffeNonlinearLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe nonlinear layer conversion.
- int appCNNInteropCaffeDropoutLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe dropout layer conversion.
- int appCNNInteropCaffeReadMean (timlCNNLayer ∗layer, const char ∗fileName)

  Read Caffe mean binary file.
- int appCNNInteropCaffePermuteMean (float ∗mean, int row, int col, int channel)

  Permute the mean in the input layer from BGR sequence to RGB.

5.4.1 Detailed Description

CNN Caffe interoperation applications.

5.4.2 Function Documentation

5.4.2.1 timlConvNeuralNetwork ∗ appCNNInteropCaffeConvert (const char ∗ *netStructurePrototxtFileName,* const char ∗ *netParamPrototxtFileName*)

Convert Caffe to TIML CNN.

Parameters

in	*netStructurePrototxtFileName*	Net structure prototxt file name
in	*netParamPrototxtFileName*	Net params prototxt file name

Returns

CNN

5.4.2.2 int appCNNInteropCaffeConvLayerConvert (timlConvNeuralNetwork ∗ *cnn,* LayerParameter *layerStructure,* LayerParameter *layerParam*)

Convert Caffe conv layer.

Parameters

in	*cnn*	CNN
in	*layerStructure*	Layer structure
in	*layerParam*	Layer params

Returns

Error code

5.4.2.3 int appCNNInteropCaffeConvLayerPermuteKernel (timlCNNLayer ∗ *layer*)

Change the kernel from BGR squence to RGB.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.4.2.4 int appCNNInteropCaffeDropoutLayerConvert (timlConvNeuralNetwork ∗ *cnn,* LayerParameter *layerStructure,* LayerParameter *layerParam*)

Caffe dropout layer conversion.

Parameters

in	*cnn*	CNN
in	*layerStructure*	Layer structure
in	*layerParam*	Layer params

Returns

Error code

5.4.2.5 int appCNNInteropCaffeFillBlockDiagonalMatrix (float ∗ *a,* int *M,* int *N,* int *group,* float ∗ *b*)

Fill a block diagonal matrix.

Parameters

out	*a*	Block diagonal matrix
in	*M*	Rows of a
in	*N*	Cols of a
in	*group*	Number of groups
in	*b*	Diagonal blocks

Returns

Error code

5.4.2.6 int appCNNInteropCaffeFlipKernelMatrix (float ∗ *kernel,* int *kernelRow,* int *kernelCol,* int *inputChannel,* int *outputChannel*)

Flip the kernels.

Parameters

in,out	*kernel*	Kernel matrix
in	*kernelRow*	Kernel rows
in	*kernelCol*	Kernel cols
in	*inputChannel*	Input feature map channels
in	*outputChannel*	Output feature map channels

Returns

Error code

5.4.2.7 int appCNNInteropCaffeFlipMatrixFloat (float ∗ *a,* int *m,* int *n*)

Flip a matrix.

Parameters

in,out	*a*	Matrix
in	*m*	Rows
in	*n*	Cols

Returns

Error code

5.4.2.8 timlCNNLayerType appCNNInteropCaffeLayerTypeConvert (LayerParameter_LayerType *type*)

Caffe to TIML CNN layer type conversion.

Parameters

in	*type*	Caffe layer type

Returns

TIML CNN layer type

5.4.2.9 int appCNNInteropCaffeLinearLayerConvert (timlConvNeuralNetwork $*$ *cnn,* LayerParameter *layerStructure,* LayerParameter *layerParam*)

Caffe linear layer conversion.

Parameters

in	*cnn*	CNN
in	*layerStructure*	Layer structure
in	*layerParam*	Layer param

Returns

Error code

5.4.2.10 int appCNNInteropCaffeNonlinearLayerConvert (timlConvNeuralNetwork $*$ *cnn,* LayerParameter *layerStructure,* LayerParameter *layerParam*)

Caffe nonlinear layer conversion.

Parameters

in	*cnn*	CNN
in	*layerStructure*	Layer structure
in	*layerParam*	Layer param

Returns

Error code

5.4.2.11 timlUtilActivationType appCNNInteropCaffeNonlinearTypeConvert (LayerParameter_LayerType *type*)

Caffe nonlinear layer type conversion.

Parameters

in	*cnn*	CNN
in	*type*	Caffe layer type

Returns

TIML CNN nonlinear layer type

5.4.2.12 int appCNNInteropCaffeNormLayerConvert (timlConvNeuralNetwork $*$ *cnn,* LayerParameter *layerStructure,* LayerParameter *layerParam*)

Caffe norm layer conversion.

Parameters

in	*cnn*	CNN
in	*layerStructure*	Layer structure
in	*layerParam*	Layer params

Returns

Error code

5.4.2.13 int appCNNInteropCaffePermuteMean (float ∗ *mean,* int *row,* int *col,* int *channel*)

Permute the mean in the input layer from BGR sequence to RGB.

Parameters

in	*mean*	Mean matrix
in	*row*	Rows
in	*col*	Cols
in	*channel*	Channels

Returns

Error code

5.4.2.14 int appCNNInteropCaffePoolingLayerConvert (timlConvNeuralNetwork ∗ *cnn,* LayerParameter *layerStructure,* LayerParameter *layerParam*)

Caffe pooling layer conversion.

Parameters

in	*cnn*	CNN
in	*layerStructure*	Layer structure
in	*layerParam*	Layer params

Returns

Error code

5.4.2.15 timlCNNPoolingType appCNNInteropCaffePoolingTypeConvert (PoolingParameter_PoolMethod *method*)

Caffe pooling type conversion.

Parameters

in	*method*	Caffe pooling method

Returns

TIML CNN pooling type

5.4.2.16 int appCNNInteropCaffeReadMean (timlCNNLayer ∗ *layer,* const char ∗ *fileName*)

Read Caffe mean binary file.

Parameters

in	*layer*	Input layer ptr
in	*fileName*	File name

Returns

Error code

5.4.2.17 bool appCNNInteropCaffeReadProtoFromBinaryFile (const char ∗ *fileName,* Message ∗ *proto*)

Caffe read proto from binary file.

Parameters

in	*fileName*	File name
in	*proto*	Proto

Returns

Status

5.4.2.18 bool appCNNInteropCaffeReadProtoFromTextFile (const char ∗ *fileName,* Message ∗ *proto*)

Caffe read proto from text file.

Parameters

in	*fileName*	File name
in	*proto*	Proto

Returns

Status

5.4.2.19 int main (int *argc,* char ∗ *argv[ ]*)

Caffe to TIML CNN model converter.

argv[0] = program name argv[1] = saved timl CNN model file name argv[2] = Caffe model text file name argv[3] = Caffe model binary file name argv[4] = Caffe mean binary file name (optional)

Returns

Error code

5.5 appCNNScene

CNN scene labeling application.

Data Structures

- struct appCNNSceneDataSet

Functions

- float appCNNSceneAccuracy (int ∗labelMatrix, int ∗trueLabelMatrix, int dim)

  Return the labeling accuracy.
- int appCNNSceneSupervisedTraining (timlConvNeuralNetwork ∗cnn, appCNNSceneDataSet ∗dataSet)

  Supervised training on the dataset.
- int appCNNSceneSBDTraining ()

  Scene labeling training example.
- int appCNNSceneSBDTesting ()

  Standford Backgournd Database Scene labeling testing example.
- int appCNNSceneClassify (timlConvNeuralNetwork ∗cnn, timlUtilImage image, int ∗labelMatrix, int scale)

  Pixel label classification.
- int appCNNSceneShuffleIdx (int ∗imageIdx, int ∗rowIdx, int ∗colIdx, appCNNSceneDataSet ∗dataSet)

  Shuffles the (image, row, col) index combination from the data set.
- int appCNNSceneGetLabel (int imageIdx, int rowIdx, int colIdx, appCNNSceneDataSet ∗dataSet)

  Return the pixel label for (image, row, col) index combination.
- int appCNNSceneGetPatch (int imageIdx, int rowIdx, int colIdx, appCNNSceneDataSet ∗dataSet, float ∗patch)

  Return the image patch for (image, row, col) index combination.
- int appCNNSceneClassifyOpenMP (timlConvNeuralNetwork ∗∗cnnTeam, int teamNum, float ∗data, int row, int col, int channel, int ∗labelMatrix, int scale)

  Supervised training on the dataset using openmp.
- int appCNNSceneLabelMatrix (float ∗map, int row, int col, int channel, int m, int k, int ∗labelMatrix, int numRow, int numCol)

  Fill the label matrix.

5.5.1 Detailed Description

CNN scene labeling application.

5.5.2 Function Documentation

5.5.2.1 float appCNNSceneAccuracy (int ∗ *labelMatrix,* int ∗ *trueLabelMatrix,* int *dim*)

Return the labeling accuracy.

Parameters

in	*labelMatrix*	Generated label matrix
in	*trueLabelMatrix*	True label matrix

in	*dim*	Dimension of the label matrix

Returns

Labeling accuracy percentage

5.5.2.2 int appCNNSceneClassify (timlConvNeuralNetwork ∗ *cnn,* timlUtilImage *image,* int ∗ *labelMatrix,* int *scale*)

Pixel label classification.

Parameters

in,out	*cnn*	CNN
in	*image*	Image
in,out	*labelMatrix*	Generated label matrix
in	*scale*	Down scaling factor of the label matrix

Returns

Error code

5.5.2.3 int appCNNSceneClassifyOpenMP (timlConvNeuralNetwork ∗∗ *cnnTeam,* int *teamNum,* float ∗ *data,* int *row,* int *col,* int *channel,* int ∗ *labelMatrix,* int *scale*)

Supervised training on the dataset using openmp.

Parameters

in,out	*cnnTeam*	An array of CNNs that share the same parameters
in	*teamNum*	Team number
in	*data*	Image data
in	*row*	Image row
in	*col*	Image col
in	*channel*	Image channel
in,out	*labelMatrix*	Generated label matrix
in	*scale*	Down scaling factor of the label matrix

Returns

Error code

5.5.2.4 int appCNNSceneGetLabel (int *imageIdx,* int *rowIdx,* int *colIdx,* appCNNSceneDataSet ∗ *dataSet*)

Return the pixel label for (image, row, col) index combination.

Parameters

in	*imageIdx*	Image index
in	*rowIdx*	Row index
in	*colIdx*	Col index
in	*dataSet*	Data set

Returns

Pixel label

5.5.2.5 int appCNNSceneGetPatch (int *imageIdx,* int *rowIdx,* int *colIdx,* appCNNSceneDataSet ∗ *dataSet,* float ∗ *patch*)

Return the image patch for (image, row, col) index combination.

Parameters

in	*imageIdx*	image index
in	*rowIdx*	row index
in	*colIdx*	col index
in	*dataSet*	data set
in,out	*patch*	image patch

Returns

Error code

5.5.2.6 int appCNNSceneLabelMatrix (float ∗ *map*, int *row*, int *col*, int *channel*, int *m*, int *k*, int ∗ *labelMatrix*, int *numRow*, int *numCol*)

Fill the label matrix.

Parameters

in	*map*	Feature map output of a CNN
in	*row*	Image row
in	*col*	Image col
in	*channel*	Image channel
in	*m*	Row Index
in	*k*	Col Index
out	*labelMatrix*	Label matrix
in	*numRow*	Num of rows
in	*numCol*	Num of cols

Returns

Error code

5.5.2.7 int appCNNSceneShuffleIdx (int ∗ *imageIdx*, int ∗ *rowIdx*, int ∗ *colIdx*, appCNNSceneDataSet ∗ *dataSet*)

Shuffles the (image, row, col) index combination from the data set.

Parameters

in,out	*imageIdx*	Image index array ptr
in,out	*rowIdx*	Row index array ptr
in,out	*colIdx*	Col index array ptr
in	*dataSet*	Data set

Returns

Error code

5.5.2.8 int appCNNSceneSupervisedTraining (timlConvNeuralNetwork ∗ *cnn*, appCNNSceneDataSet ∗ *dataSet*)

Supervised training on the dataset.

Parameters

in,out	*cnn*	CNN
in	*dataSet*	Data set

Returns

Error code

5.6 benchmarkCNNClass

Benchmark CNN classification.

Functions

- int benchmarkCNNClassCaffeNetTesting ()

    CNN CaffeNet classification benchmark.
- int benchmarkCNNClassVGGNetTesting ()

    CNN VGGNet classification benchmark.

5.6.1 Detailed Description

Benchmark CNN classification.

5.7 cnn

Convolutional neural network.

Data Structures

- struct timlCNNPoolingParams
- struct timlCNNLinearParams
- struct timlCNNDataSet
- struct timlCNNConvParams
- struct timlCNNNonlinearParams
- struct timlCNNNormParams
- struct timlCNNInputParams
- struct timlCNNDropoutParams
- struct _timlCNNLayer_
- struct timlCNNTrainingParams
- struct _timlConvNeuralNetwork_

Macros

- #define **ERROR_CNN_OFFSET** 4000

Typedefs

- typedef struct _timlCNNLayer_ **timlCNNLayer**
- typedef struct _timlConvNeuralNetwork_ **timlConvNeuralNetwork**

Enumerations

- enum **timlCNNError** {
  ERROR_CNN_FEATURE_MAP_SIZE = ERROR_CNN_OFFSET, **ERROR_CNN_FEATURE_MAP_CHANNEL**, **ERROR_CNN_ALLOCATION**, **ERROR_CNN_LAYER_ALLOCATION**,
  ERROR_CNN_TEAM_ALLOCATION, **ERROR_CNN_CONV_LAYER_KERNEL_SIZE**, **ERROR_CNN_CONV_LAYER_PAD_SIZE**, **ERROR_CNN_CONV_LAYER_STRIDE_SIZE**,
  ERROR_CNN_POOLING_LAYER_SCALE_SIZE, **ERROR_CNN_POOLING_LAYER_PAD_SIZE**, **ERROR_CNN_POOLING_LAYER_STRIDE_SIZE**, **ERROR_CNN_INPUT_LAYER_PARAMS**,
  ERROR_CNN_LINEAR_LAYER_DIM, **ERROR_CNN_NORM_LAYER_PARAMS**, **ERROR_CNN_DROPOUT_LAYER_PARAMS**, **ERROR_CNN_NULL_PTR**,
  ERROR_CNN_EMPTY, **ERROR_CNN_READ_FILE**, **ERROR_CNN_CLASS** }
- enum **timlCNNNormType** { **CNN_InterChannel**, **CNN_IntraChannel** }
- enum **timlCNNLayerType** {
  CNN_Input, **CNN_Conv**, **CNN_Pooling**, **CNN_Nonlinear**,
  CNN_Linear, **CNN_Norm**, **CNN_Dropout** }
- enum **timlCNNPoolingType** { **CNN_MaxPooling**, **CNN_MeanPooling** }

Functions

- int timlCNNInputShareParams (timlConvNeuralNetwork ∗cnn, timlCNNLayer ∗layer)

  Share the mean with other input layer.
- int timlCNNConvShareParams (timlConvNeuralNetwork ∗cnnShare, timlCNNLayer ∗layer)

  Share the parameters with other conv layer.

- int timlCNNLinearShareParams (timlConvNeuralNetwork ∗cnnShare, timlCNNLayer ∗layer)

    Share the parameters with other linear layer.
- int timlCNNConvInitialize (timlCNNLayer ∗layer)

    Initialize the conv layer.
- int timlCNNLinearInitialize (timlCNNLayer ∗layer)

    Initialize the linear layer.
- int timlCNNNonlinearInitialize (timlCNNLayer ∗layer)

    Initialize the nonlinear layer.
- int timlCNNInputInitialize (timlCNNLayer ∗layer)

    Initialize the input layer.
- int timlCNNPoolingInitialize (timlCNNLayer ∗layer)

    Initialize the pooling layer.
- int timlCNNNormInitialize (timlCNNLayer ∗layer)

    Initialize the norm layer.
- int timlCNNDropoutInitialize (timlCNNLayer ∗layer)

    Initialize the dropout layer.
- int timlCNNBackPropagation (timlConvNeuralNetwork ∗cnn, timlCNNLayer ∗layer)

    Back propagate the gradient from layer to the first layer of the cnn.
- int timlCNNConvBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the conv layer to the previous layer.
- int timlCNNNormBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the norm layer to the previous layer.
- int timlCNNPoolingBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the pooling layer to the previous layer.
- int timlCNNMaxPoolingBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the max pooling layer to the previous layer.
- int timlCNNMeanPoolingBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the mean pooling layer to the previous layer.
- int timlCNNNonlinearBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the nonlinear layer to the previous layer.
- int timlCNNLinearBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the linear layer to the previous layer.
- int timlCNNDropoutBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the dropout layer to the previous layer.
- int timlCNNCostWithLabel (timlConvNeuralNetwork ∗cnn, int label, float ∗cost, timlCNNLayer ∗∗bpStartLayer)

    Calculate the cost based on the cnn output and the label.
- int timlCNNForwardPropagation (timlConvNeuralNetwork ∗cnn, float ∗data, int dim)

    Forward propagate data to the CNN.
- int timlCNNInputForwardPropagation (timlCNNLayer ∗layer, float ∗data, int dim)

    Forward propagate data to the the input layer.
- int timlCNNLinearForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNDropoutForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNNonlinearForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNNormForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNPoolingForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.

- int timlCNNMaxPoolingForwardPropagation (timlCNNLayer ∗prevLayer)

  Forward propagate form layer to layer->next.
- int timlCNNMeanPoolingForwardPropagation (timlCNNLayer ∗prevLayer)

  Forward propagate form layer to layer->next.
- int timlCNNConvForwardPropagation (timlCNNLayer ∗prevLayer)

  Forward propagate form layer to layer->next.
- int timlCNNDeleteConvLayer (timlCNNLayer ∗layer)

  Delete conv layer.
- int timlCNNDeleteInputLayer (timlCNNLayer ∗layer)

  Delete input layer.
- int timlCNNDeleteNonlinearLayer (timlCNNLayer ∗layer)

  Delete nonlinear layer.
- int timlCNNDeleteNormLayer (timlCNNLayer ∗layer)

  Delete norm layer.
- int timlCNNDeletePoolingLayer (timlCNNLayer ∗layer)

  Delete pooling layer.
- int timlCNNDeleteLinearLayer (timlCNNLayer ∗layer)

  Delete linear layer.
- int timlCNNDeleteDropoutLayer (timlCNNLayer ∗layer)

  Delete dropout layer.
- int timlCNNResetConvLayer (timlCNNLayer ∗layer)

  Reset conv layer.
- int timlCNNResetInputLayer (timlCNNLayer ∗layer)

  Reset input layer.
- int timlCNNResetLinearLayer (timlCNNLayer ∗layer)

  Reset linear layer.
- int timlCNNResetNonlinearLayer (timlCNNLayer ∗layer)

  Reset nonlinear layer.
- int timlCNNResetNormLayer (timlCNNLayer ∗layer)

  Reset norm layer.
- int timlCNNResetPoolingLayer (timlCNNLayer ∗layer)

  Reset pooling layer.
- int timlCNNUpdateParams (timlConvNeuralNetwork ∗cnn)

  Update the parameters of the cnn.
- int timlCNNLinearUpdateParams (timlCNNLayer ∗layer)

  Update the parameters of the linear layer.
- int timlCNNConvUpdateParams (timlCNNLayer ∗layer)

  Update the parameters of the conv layer.
- int timlCNNConvWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the conv layer to file(s)
- int timlCNNNonlinearWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the nonlinear layer to file(s)
- int timlCNNNormWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the norm layer to file(s)
- int timlCNNPoolingWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the pooling layer to file(s)

- int timlCNNLinearWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the linear layer to file(s)
- int timlCNNInputWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the input layer to file(s)
- int timlCNNTrainingParamsWriteToFile (FILE ∗fp, timlConvNeuralNetwork ∗cnn, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the training params to file(s)
- int timlCNNDropoutWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the dropout layer to file(s)
- int timlCNNConvReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the conv layer from a Matlab compatible text file.
- int timlCNNTrainingParamsReadFromTextFile (FILE ∗fp, timlConvNeuralNetwork ∗cnn)

  Read the training params from a text file.
- int timlCNNNormReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the nonlinear layer from a text file.
- int timlCNNPoolingReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the pooling layer from a text file.
- int timlCNNNonlinearReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the nonlinear layer from a text file.
- int timlCNNLinearReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the linear layer from a text file.
- int timlCNNDropoutReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the dropout layer from a text file.
- int timlCNNInputReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the input layer from a Matlab compatible text file.
- int timlCNNConvReadFromBinaryFile (FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer)

  Read the conv layer parameters from binary files.
- int timlCNNLinearReadFromBinaryFile (FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer)

  Read the linear layer parameters from binary files.
- int timlCNNInputReadFromBinaryFile (FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer)

  Read the input layer parameters from binary files.
- int timlCNNAssignDevice (timlConvNeuralNetwork ∗cnn, int deviceId, int threadId)

  Assign the cnn to a specific device and thread.
- const char ∗ timlCNNLayerTypeStr (timlCNNLayer ∗layer)

  Return a string that represents the layer type.
- int timlCNNMemPoolSize (timlConvNeuralNetwork ∗cnn)

  Return the memory pool size (byte)
- int timlCNNSupervisedTrainingWithLabelBatchModeOpenMP (timlConvNeuralNetwork ∗cnn, float ∗data, int ∗label, int dim, int num)

  supervised training with label using openmp
- int timlCNNClassifyTopNBatchModeOpenMP (timlConvNeuralNetwork ∗cnn, float ∗data, int dim, int num, int ∗label, float ∗percent, int topN)

  Batch classification using openmp.
- int timlCNNClassifyTopNTeamModeOpenMP (timlConvNeuralNetwork ∗∗cnnTeam, int num, float ∗data, int dim, int ∗label, float ∗percent, int topN)

  Batch classification using openmp.
- int timlCNNAddConvLayer (timlConvNeuralNetwork ∗cnn, int kernelRow, int kernelCol, int strideX, int strideY, int featureMapChannel, timlCNNConvParams params)

  Add conv layer.

- int timlCNNAddDropoutLayer (timlConvNeuralNetwork ∗cnn, float prob)

  Add dropout layer.
- int timlCNNAddInputLayer (timlConvNeuralNetwork ∗cnn, int featureMapRow, int featureMapCol, int featureMapChannel, timlCNNInputParams params)

  Add input layer.
- int timlCNNAddLinearLayer (timlConvNeuralNetwork ∗cnn, int dim, timlCNNLinearParams params)

  Add linear layer.
- int timlCNNAddNonlinearLayer (timlConvNeuralNetwork ∗cnn, timlUtilActivationType type)

  Add nonlinear layer.
- int timlCNNAddNormLayer (timlConvNeuralNetwork ∗cnn, timlCNNNormParams params)

  Add normalization layer.
- int timlCNNAddPoolingLayer (timlConvNeuralNetwork ∗cnn, int scaleRow, int scaleCol, int strideX, int strideY, timlCNNPoolingType type, timlCNNPoolingParams params)

  Add pooling layer.
- int timlCNNClassifyTop1SingleMode (timlConvNeuralNetwork ∗cnn, float ∗data, int dim)

  Classify the data.
- int timlCNNClassifyTopNBatchMode (timlConvNeuralNetwork ∗cnn, float ∗data, int dim, int num, int ∗label, float ∗percent, int topN)

  Batch classification.
- timlConvNeuralNetwork ∗ timlCNNClone (timlConvNeuralNetwork ∗cnn, int deviceId)

  Clone a cnn.
- timlCNNConvParams timlCNNConvParamsDefault ()

  Return the default parameters for the convolutional layer.
- timlConvNeuralNetwork ∗ timlCNNCreateConvNeuralNetwork (timlCNNTrainingParams params, int deviceId)

  Create a cnn structure.
- int timlCNNDelete (timlConvNeuralNetwork ∗cnn)

  Free a cnn structure.
- int timlCNNGetLayerNum (timlConvNeuralNetwork ∗cnn)

  Return the number of layers of the cnn.
- long timlCNNGetParamsNum (timlConvNeuralNetwork ∗cnn)

  Get the number of parameters of the cnn.
- int timlCNNInitialize (timlConvNeuralNetwork ∗cnn)

  Allocate the memory required by the cnn.
- timlCNNLinearParams timlCNNLinearParamsDefault ()

  Return the default parameters for the linear layer.
- long timlCNNMemory (timlConvNeuralNetwork ∗cnn)

  Return the memory in bytes required by the cnn.
- timlCNNNonlinearParams timlCNNNonlinearParamsDefault ()

  Return the default parameters for the nonlinear layer.
- timlCNNNormParams timlCNNNormParamsDefault ()

  Return the default parameters for the norm layer.
- timlCNNPoolingParams timlCNNPoolingParamsDefault ()

  Return the default parameters for the pooling layer.
- int timlCNNPrint (timlConvNeuralNetwork ∗cnn)

  Print out the information of the cnn.
- int timlCNNProfile (timlConvNeuralNetwork ∗cnn, float ∗data, int dim, int num, int ∗label, int iter)

  Profile the CNN with both timing and memory allocation.
- timlConvNeuralNetwork ∗ timlCNNReadFromFile (const char ∗fileName, int deviceId)

  Read CNN from file(s)
- int timlCNNReset (timlConvNeuralNetwork ∗cnn)

  Reset the parameters of the CNN.

- int timlCNNResetDropoutLayer (timlCNNLayer ∗layer)

  Reset dropout layer.
- int timlCNNResize (timlConvNeuralNetwork ∗cnn, int row, int col, int channel)

  Resize the feature map sizes to accommodate new input feature map dimensions.
- int timlCNNSetMode (timlConvNeuralNetwork ∗cnn, timlUtilPhase phase)

  Set the phase (train/test) of the cnn.
- timlConvNeuralNetwork ∗ timlCNNShareParams (timlConvNeuralNetwork ∗cnn, int deviceId)

  Create a new CNN that shares the parameters with the input CNN.
- int timlCNNSupervisedTrainingWithLabelBatchMode (timlConvNeuralNetwork ∗cnn, float ∗data, int ∗label, int dim, int num)

  Supervised training with label.
- timlCNNTrainingParams timlCNNTrainingParamsDefault ()

  Return the default training parameters.
- int timlCNNWriteToFile (const char ∗fileName, timlConvNeuralNetwork ∗cnn, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the cnn to file(s)

- timlCNNInputParams timlCNNInputParamsDefault ()

  Return the default parameters for the input layer.

5.7.1 Detailed Description

Convolutional neural network.

5.7.2 Function Documentation

5.7.2.1 int timlCNNAddConvLayer (**timlConvNeuralNetwork** ∗ *cnn,* int *kernelRow,* int *kernelCol,* int *strideX,* int *strideY,* int *featureMapChannel,* **timlCNNConvParams** *params*)

Add conv layer.

Parameters

`in,out`	*cnn*	CNN
`in`	*kernelRow*	Kernel row size
`in`	*kernelCol*	Kernel col size
`in`	*strideX*	Kernel horizontal stride size
`in`	*strideY*	Kernel vertical stride size
`in`	*featureMap-Channel*	Output feature map channel size
`in`	*params*	Optional parameters

Returns

Error code

5.7.2.2 int timlCNNAddDropoutLayer (**timlConvNeuralNetwork** ∗ *cnn,* float *prob*)

Add dropout layer.

Parameters

in,out	*cnn*	CNN
in	*prob*	Dropout probability

Returns

Error code

5.7.2.3 int timlCNNAddInputLayer (timlConvNeuralNetwork ∗ *cnn*, int *featureMapRow*, int *featureMapCol*, int *featureMapChannel*, timlCNNInputParams *params*)

Add input layer.

Parameters

in,out	*cnn*	CNN
in	*featureMapRow*	Output feature map row size
in	*featureMapCol*	Output feature map col size
in	*featureMapChannel*	Output feature map channel size
in	*params*	Optional parameters

Returns

Error code

5.7.2.4 int timlCNNAddLinearLayer (timlConvNeuralNetwork ∗ *cnn*, int *dim*, timlCNNLinearParams *params*)

Add linear layer.

Parameters

in,out	*cnn*	CNN
in	*dim*	Output 1D feature map dimension
in	*params*	Optional parameters

Returns

Error code

5.7.2.5 int timlCNNAddNonlinearLayer (timlConvNeuralNetwork ∗ *cnn*, timlUtilActivationType *type*)

Add nonlinear layer.

Parameters

in,out	*cnn*	CNN
in	*type*	Nonlinear activation type

Returns

Error code

5.7.2.6 int timlCNNAddNormLayer (timlConvNeuralNetwork ∗ *cnn*, timlCNNNormParams *params*)

Add normalization layer.

Parameters

in,out	*cnn*	CNN
in	*params*	Optional parameters

Returns

Error code

5.7.2.7 int timlCNNAddPoolingLayer (timlConvNeuralNetwork ∗ *cnn,* int *scaleRow,* int *scaleCol,* int *strideX,* int *strideY,* timlCNNPoolingType *type,* timlCNNPoolingParams *params*)

Add pooling layer.

Parameters

in,out	*cnn*	CNN
in	*scaleRow*	Scale kernel row size
in	*scaleCol*	Scale kernel col size
in	*strideX*	Scale kernel horizontal stride size
in	*strideY*	Scale kernel vertical stride size
in	*type*	Pooling type (max/mean)
in	*params*	Optional parameters

Returns

Error code

5.7.2.8 int timlCNNAssignDevice (timlConvNeuralNetwork ∗ *cnn,* int *deviceId,* int *threadId*)

Assign the cnn to a specific device and thread.

Parameters

in	*cnn*	CNN
in	*deviceId*	Device Id starting from 0;
in	*threadId*	Thread Id starting from 0;

Returns

Error code

5.7.2.9 int timlCNNBackPropagation (timlConvNeuralNetwork ∗ *cnn,* timlCNNLayer ∗ *layer*)

Back propagate the gradient from layer to the first layer of the cnn.

Parameters

in	*cnn*	CNN
in	*layer*	Start layer

Returns

Error code

5.7.2.10 int timlCNNClassifyTop1SingleMode (timlConvNeuralNetwork ∗ *cnn,* float ∗ *data,* int *dim*)

Classify the data.

Parameters

in,out	*cnn*	CNN
in	*data*	Data
in	*dim*	Dimension of the data

Returns

Label

5.7.2.11 int timlCNNClassifyTopNBatchMode (**timlConvNeuralNetwork** $*$ *cnn,* float $*$ *data,* int *dim,* int *num,* int $*$ *label,* float $*$ *percent,* int *topN*)

Batch classification.

Parameters

in,out	*cnn*	CNN
in	*data*	Data batch
in	*dim*	Data dimension
in	*num*	Data number
out	*label*	Label array ptr, size = num$*$topN
out	*percent*	Percent array ptr, size = num$*$topN
out	*topN*	Output the top N labels and the corresponding percentage

Returns

Error code

5.7.2.12 int timlCNNClassifyTopNBatchModeOpenMP (**timlConvNeuralNetwork** $*$ *cnn,* float $*$ *data,* int *dim,* int *num,* int $*$ *label,* float $*$ *percent,* int *topN*)

Batch classification using openmp.

Parameters

in,out	*cnn*	CNN
in	*data*	Data batch
in	*dim*	Data dimension
in	*num*	Data number
out	*label*	Label array ptr, size = num$*$topN
out	*percent*	Percent array ptr, size = num$*$topN
out	*topN*	Output the top N labels and the corresponding percentage

Returns

Error code

5.7.2.13 int timlCNNClassifyTopNTeamModeOpenMP (**timlConvNeuralNetwork** $**$ *cnnTeam,* int *num,* float $*$ *data,* int *dim,* int $*$ *label,* float $*$ *percent,* int *topN*)

Batch classification using openmp.

This is the same function as timlCNNBatchClassifyOpenMP but avoids creating and deleting the cnn team each time the function is called

Parameters

in,out	*cnnTeam*	An array of CNNs that shares the same parameters
in	*num*	Size of the CNN array as well as the data
in	*data*	Data batch
in	*dim*	Data dimension
in,out	*label*	Label array ptr, size = num∗topN
in,out	*percent*	Percent array ptr, size = num∗topN
in,out	*topN*	Output the top N labels and the corresponding percentage

Returns

Error code

5.7.2.14 timlConvNeuralNetwork∗ timlCNNClone (timlConvNeuralNetwork ∗ *cnn,* int *deviceId*)

Clone a cnn.

Parameters

in	*cnn*	CNN to be cloned
in	*deviceId*	Device Id

Returns

Cloned cnn

5.7.2.15 int timlCNNConvBackPropagation (timlCNNLayer ∗ *layer*)

Back propagate the gradient from the conv layer to the previous layer.

layer->prev->delta[i] = sum_{j}(layer->delta[j] conv2full layer->kernel[i, j])

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.16 int timlCNNConvForwardPropagation (timlCNNLayer ∗ *prevLayer*)

Forward propagate form layer to layer->next.

Parameters

in	*prevLayer*	Previous layer ptr

Returns

Error code

5.7.2.17 int timlCNNConvInitialize (timlCNNLayer ∗ *layer*)

Initialize the conv layer.

Parameters

`in`	*layer*	Conv layer

Returns

Error code

5.7.2.18 timlCNNConvParams timlCNNConvParamsDefault ( )

Return the default parameters for the convolutional layer.

Returns

Default conv layer parameters

5.7.2.19 int timlCNNConvReadFromBinaryFile (FILE ∗ *fp2*, FILE ∗ *fp3*, timlCNNLayer ∗ *layer*)

Read the conv layer parameters from binary files.

Parameters

`in`	*fp2*	FILE ptr to the level 2 parameter bin file
`in`	*fp3*	FILE ptr to the level 3 state bin file
`in,out`	*layer*	Conv layer

Returns

Error code

5.7.2.20 int timlCNNConvReadFromTextFile (FILE ∗ *fp1*, timlConvNeuralNetwork ∗ *cnn*)

Read the conv layer from a Matlab compatible text file.

Parameters

`in`	*fp1*	FILE ptr to the level 1 text file
`in,out`	*cnn*	CNN

Returns

Error code

5.7.2.21 int timlCNNConvShareParams (timlConvNeuralNetwork ∗ *cnnShare*, timlCNNLayer ∗ *layer*)

Share the parameters with other conv layer.

Add a conv layer to cnnShare that shares the same parameters as the conv layer

Parameters

`in,out`	*cnnShare*	CNN that shares the parameters of other cnn

in	*layer*	Target cnn layer to share its parameters

Returns

Error code

5.7.2.22 int timlCNNConvUpdateParams (**timlCNNLayer** ∗ *layer*)

Update the parameters of the conv layer.

Parameters

in	*layer*	Layer

Returns

Error code

5.7.2.23 int timlCNNConvWriteToFile (FILE ∗ *fp1,* FILE ∗ *fp2,* FILE ∗ *fp3,* **timlCNNLayer** ∗ *layer,* **timlUtilParamsLevel** *level,* const char ∗ *name,* const char ∗ *floatFormat,* const char ∗ *intFormat*)

Write the conv layer to file(s)

Parameters

in,out	*fp1*	FILE ptr to the level 1 text file
in,out	*fp2*	FILE ptr to the level 2 bin file
in,out	*fp3*	FILE ptr to the level 3 bin file
in	*layer*	Layer ptr
in	*level*	Write level
in	*name*	CNN name
in	*floatFormat*	Format string for floats
in	*intFormat*	Format string for ints

Returns

Error code

5.7.2.24 int timlCNNCostWithLabel (**timlConvNeuralNetwork** ∗ *cnn,* int *label,* float ∗ *cost,* **timlCNNLayer** ∗∗ *bpStartLayer*)

Calculate the cost based on the cnn output and the label.

Parameters

in	*cnn*	CNN
in	*label*	Label
in,out	*cost*	Cost
in,out	*bpStartLayer*	Back propagation start layer

Returns

Error code

5.7.2.25 timlConvNeuralNetwork∗ timlCNNCreateConvNeuralNetwork (timlCNNTrainingParams *params,* int *deviceId*)

Create a cnn structure.

Parameters

`in`	*params*	Training parameters
`in`	*deviceId*	Device Id, the default value is 0

Returns

CNN

5.7.2.26 int timlCNNDelete (timlConvNeuralNetwork ∗ *cnn*)

Free a cnn structure.

Parameters

`in`	*cnn*	CNN structure

Returns

Error code

5.7.2.27 int timlCNNDeleteConvLayer (timlCNNLayer ∗ *layer*)

Delete conv layer.

Parameters

`in`	*layer*	Layer ptr

Returns

Error code

5.7.2.28 int timlCNNDeleteDropoutLayer (timlCNNLayer ∗ *layer*)

Delete dropout layer.

Parameters

`in`	*layer*	Layer ptr

Returns

Error code

5.7.2.29 int timlCNNDeleteInputLayer (timlCNNLayer ∗ *layer*)

Delete input layer.

Parameters

`in`	*layer*	Layer ptr

Returns

Error code

5.7.2.30 int timlCNNDeleteLinearLayer (**timlCNNLayer** ∗ *layer*)

Delete linear layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.31 int timlCNNDeleteNonlinearLayer (**timlCNNLayer** ∗ *layer*)

Delete nonlinear layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.32 int timlCNNDeleteNormLayer (**timlCNNLayer** ∗ *layer*)

Delete norm layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.33 int timlCNNDeletePoolingLayer (**timlCNNLayer** ∗ *layer*)

Delete pooling layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.34 int timlCNNDropoutBackPropagation (**timlCNNLayer** ∗ *layer*)

Back propagate the gradient from the dropout layer to the previous layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.35 int timlCNNDropoutForwardPropagation (timlCNNLayer * *prevLayer*)

Forward propagate form layer to layer->next.

Parameters

in	*prevLayer*	Previous layer ptr

Returns

Error code

5.7.2.36 int timlCNNDropoutInitialize (**timlCNNLayer** ∗ *layer*)

Initialize the dropout layer.

Parameters

in	*layer*	Dropout layer

Returns

Error code

5.7.2.37 int timlCNNDropoutReadFromTextFile (FILE ∗ *fp1*, **timlConvNeuralNetwork** ∗ *cnn*)

Read the dropout layer from a text file.

Parameters

in	*fp1*	FILE ptr to the level 1 text file
in,out	*cnn*	CNN

Returns

Error code

5.7.2.38 int timlCNNDropoutWriteToFile (FILE ∗ *fp1*, FILE ∗ *fp2*, FILE ∗ *fp3*, **timlCNNLayer** ∗ *layer*, **timlUtilParamsLevel** *level*, const char ∗ *name*, const char ∗ *floatFormat*, const char ∗ *intFormat*)

Write the dropout layer to file(s)

Parameters

in,out	*fp1*	FILE ptr to the level 1 text file
in,out	*fp2*	FILE ptr to the level 2 bin file
in,out	*fp3*	FILE ptr to the level 3 bin file
in	*layer*	Layer
in	*level*	Write level
in	*name*	CNN name
in	*floatFormat*	Format string for floats
in	*intFormat*	Format string for ints

Returns

Error code

5.7.2.39 int timlCNNForwardPropagation (**timlConvNeuralNetwork** ∗ *cnn*, float ∗ *data*, int *dim*)

Forward propagate data to the CNN.

Parameters

in,out	*cnn*	
in	*data*	Data ptr
in	*dim*	Data dimension

Returns

Error code

5.7.2.40 int timlCNNGetLayerNum (timlConvNeuralNetwork ∗ *cnn*)

Return the number of layers of the cnn.

Parameters

in	*cnn*	

Returns

Layer number

5.7.2.41 long timlCNNGetParamsNum (timlConvNeuralNetwork ∗ *cnn*)

Get the number of parameters of the cnn.

Parameters

in	*cnn*	CNN

5.7.2.42 int timlCNNInitialize (timlConvNeuralNetwork ∗ *cnn*)

Allocate the memory required by the cnn.

Parameters

in	*cnn*	CNN

Returns

Error code

5.7.2.43 int timlCNNInputForwardPropagation (timlCNNLayer ∗ *layer,* float ∗ *data,* int *dim*)

Forward propagate data to the the input layer.

Parameters

in	*layer*	Layer ptr
in	*data*	Data ptr
in	*dim*	Data dimension

Returns

Error code

5.7.2.44 int timlCNNInputInitialize (timlCNNLayer ∗ *layer*)

Initialize the input layer.

Parameters

in	*layer*	Input layer

Returns

Error code

5.7.2.45 timlCNNInputParams timlCNNInputParamsDefault ( )

Return the default parameters for the input layer.

Returns

Default input layer parameters

5.7.2.46 int timlCNNInputReadFromBinaryFile (FILE * *fp2,* FILE * *fp3,* timlCNNLayer * *layer*)

Read the input layer parameters from binary files.

Parameters

in	*fp2*	FILE ptr to the level 2 parameter bin file
in	*fp3*	FILE ptr to the level 3 state bin file
in,out	*layer*	Input layer

Returns

Error code

5.7.2.47 int timlCNNInputReadFromTextFile (FILE * *fp1,* timlConvNeuralNetwork * *cnn*)

Read the input layer from a Matlab compatible text file.

Parameters

in	*fp1*	FILE ptr to the level 1 text file
in,out	*cnn*	CNN

Returns

Error code

5.7.2.48 int timlCNNInputShareParams (timlConvNeuralNetwork * *cnn,* timlCNNLayer * *layer*)

Share the mean with other input layer.

Add a layer who shares the same mean as the input layer to cnn

Parameters

in,out	*cnn*	CNN

in	*layer*	Layer to share its mean

Returns

Error code

5.7.2.49 int timlCNNInputWriteToFile (FILE ∗ *fp1,* FILE ∗ *fp2,* FILE ∗ *fp3,* timlCNNLayer ∗ *layer,* timlUtilParamsLevel *level,* const char ∗ *name,* const char ∗ *floatFormat,* const char ∗ *intFormat*)

Write the input layer to file(s)

Parameters

in,out	*fp1*	FILE ptr to the level 1 text file
in,out	*fp2*	FILE ptr to the level 2 bin file
in,out	*fp3*	FILE ptr to the level 3 bin file
in	*layer*	Layer ptr
in	*level*	Write level
in	*name*	CNN name
in	*floatFormat*	Format string for floats
in	*intFormat*	Format string for ints

Returns

Error code

5.7.2.50 const char ∗ timlCNNLayerTypeStr (timlCNNLayer ∗ *layer*)

Return a string that represents the layer type.

Parameters

in	*layer*	Layer pointer

Returns

Layer type string

< return unknown type

5.7.2.51 int timlCNNLinearBackPropagation (timlCNNLayer ∗ *layer*)

Back propagate the gradient from the linear layer to the previous layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.52 int timlCNNLinearForwardPropagation (timlCNNLayer ∗ *prevLayer*)

Forward propagate form layer to layer->next.

Parameters

in	*prevLayer*	Previous layer ptr

Returns

Error code

5.7.2.53 int timlCNNLinearInitialize (**timlCNNLayer** ∗ *layer*)

Initialize the linear layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.54 **timlCNNLinearParams** timlCNNLinearParamsDefault ( )

Return the default parameters for the linear layer.

Returns

Default linear layer parameters

5.7.2.55 int timlCNNLinearReadFromBinaryFile (FILE ∗ *fp2,* FILE ∗ *fp3,* **timlCNNLayer** ∗ *layer*)

Read the linear layer parameters from binary files.

Parameters

in	*fp2*	FILE ptr to the level 2 parameter bin file
in	*fp3*	FILE ptr to the level 3 state bin file
in,out	*layer*	Layer ptr

Returns

Error code

5.7.2.56 int timlCNNLinearReadFromTextFile (FILE ∗ *fp1,* **timlConvNeuralNetwork** ∗ *cnn*)

Read the linear layer from a text file.

Parameters

in	*fp1*	FILE ptr to the level 1 text file
in,out	*cnn*	CNN

Returns

Error code

5.7.2.57 int timlCNNLinearShareParams (**timlConvNeuralNetwork** ∗ *cnnShare,* **timlCNNLayer** ∗ *layer*)

Share the parameters with other linear layer.

Add a layer to cnnShare that shares the same parameters as the linear layer

Parameters

in,out	*cnnShare*	CNN that shares the same paramters with other cnn
in	*layer*	Target linear layer to share its parameters

Returns

Error code

5.7.2.58 int timlCNNLinearUpdateParams (timlCNNLayer ∗ *layer*)

Update the parameters of the linear layer.

Parameters

in,out	*layer*	Layer ptr

Returns

Error code

5.7.2.59 int timlCNNLinearWriteToFile (FILE ∗ *fp1,* FILE ∗ *fp2,* FILE ∗ *fp3,* timlCNNLayer ∗ *layer,* timlUtilParamsLevel *level,* const char ∗ *name,* const char ∗ *floatFormat,* const char ∗ *intFormat*)

Write the linear layer to file(s)

Parameters

in,out	*fp1*	FILE ptr to the level 1 text file
in,out	*fp2*	FILE ptr to the level 2 bin file
in,out	*fp3*	FILE ptr to the level 3 bin file
in	*layer*	Layer ptr
in	*level*	Write level
in	*name*	CNN name
in	*floatFormat*	Format string for floats
in	*intFormat*	Format string for ints

Returns

Error code

5.7.2.60 int timlCNNMaxPoolingBackPropagation (timlCNNLayer ∗ *layer*)

Back propagate the gradient from the max pooling layer to the previous layer.

Parameters

in,out	*layer*	Layer ptr

Returns

Error code

5.7.2.61 int timlCNNMaxPoolingForwardPropagation (timlCNNLayer ∗ *prevLayer*)

Forward propagate form layer to layer->next.

Parameters

in	*prevLayer*	Previous layer ptr

Returns

Error code

5.7.2.62 int timlCNNMeanPoolingBackPropagation (timlCNNLayer ∗ *layer*)

Back propagate the gradient from the mean pooling layer to the previous layer.

Parameters

in,out	*layer*	

Returns

Error code

5.7.2.63 int timlCNNMeanPoolingForwardPropagation (timlCNNLayer ∗ *prevLayer*)

Forward propagate form layer to layer->next.

Parameters

in,out	*prevLayer*	Previous layer

Returns

error code

5.7.2.64 long timlCNNMemory (timlConvNeuralNetwork ∗ *cnn*)

Return the memory in bytes required by the cnn.

Parameters

in	*cnn*	CNN

Returns

Required memory in byte

5.7.2.65 int timlCNNMemPoolSize (timlConvNeuralNetwork ∗ *cnn*)

Return the memory pool size (byte)

Returns

Memory pool size (byte)

5.7.2.66 int timlCNNNonlinearBackPropagation (timlCNNLayer ∗ *layer*)

Back propagate the gradient from the nonlinear layer to the previous layer.

Parameters

`in,out`	*layer*	Layer ptr

Returns

Error code

5.7.2.67 int timlCNNNonlinearForwardPropagation (**timlCNNLayer** ∗ *prevLayer*)

Forward propagate form layer to layer->next.

Parameters

`in`	*prevLayer*	Previous layer

Returns

Error code

5.7.2.68 int timlCNNNonlinearInitialize (**timlCNNLayer** ∗ *layer*)

Initialize the nonlinear layer.

Parameters

`in`	*layer*	Layer ptr

Returns

Error code

5.7.2.69 **timlCNNNonlinearParams** timlCNNNonlinearParamsDefault ( )

Return the default parameters for the nonlinear layer.

Returns

Default nonlinear layer parameters

5.7.2.70 int timlCNNNonlinearReadFromTextFile (FILE ∗ *fp1*, **timlConvNeuralNetwork** ∗ *cnn*)

Read the nonlinear layer from a text file.

Parameters

`in`	*fp1*	FILE ptr to the level 1 text file
`in,out`	*cnn*	CNN

Returns

Error code

5.7.2.71 int timlCNNNonlinearWriteToFile (FILE ∗ *fp1*, FILE ∗ *fp2*, FILE ∗ *fp3*, **timlCNNLayer** ∗ *layer*, **timlUtilParamsLevel** *level*, const char ∗ *name*, const char ∗ *floatFormat*, const char ∗ *intFormat*)

Write the nonlinear layer to file(s)

Parameters

in,out	*fp1*	FILE ptr to the level 1 text file
in,out	*fp2*	FILE ptr to the level 2 bin file
in,out	*fp3*	FILE ptr to the level 3 bin file
in	*layer*	Layer ptr
in	*level*	Write level
in	*name*	CNN name
in	*floatFormat*	Format string for floats
in	*intFormat*	Format string for ints

Returns

Error code

5.7.2.72 int timlCNNNormBackPropagation (timlCNNLayer ∗ *layer*)

Back propagate the gradient from the norm layer to the previous layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.73 int timlCNNNormForwardPropagation (timlCNNLayer ∗ *prevLayer*)

Forward propagate form layer to layer->next.

Parameters

in	*prevLayer*	Previous layer

Returns

Error code

5.7.2.74 int timlCNNNormInitialize (timlCNNLayer ∗ *layer*)

Initialize the norm layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.75 timlCNNNormParams timlCNNNormParamsDefault ( )

Return the default parameters for the norm layer.

Returns

Default norm layer parameters

5.7.2.76 int timlCNNNormReadFromTextFile (FILE ∗ *fp1,* timlConvNeuralNetwork ∗ *cnn*)

Read the nonlinear layer from a text file.

Parameters

in	*fp1*	FILE ptr to the level 1 text file
in,out	*cnn*	CNN

Returns

Error code

5.7.2.77 int timlCNNNormWriteToFile (FILE ∗ *fp1,* FILE ∗ *fp2,* FILE ∗ *fp3,* timlCNNLayer ∗ *layer,* timlUtilParamsLevel *level,* const char ∗ *name,* const char ∗ *floatFormat,* const char ∗ *intFormat*)

Write the norm layer to file(s)

Parameters

in,out	*fp1*	FILE ptr to the level 1 text file
in,out	*fp2*	FILE ptr to the level 2 bin file
in,out	*fp3*	FILE ptr to the level 3 bin file
in	*layer*	Layer ptr
in	*level*	Write level
in	*name*	CNN name
in	*floatFormat*	format string for floats
in	*intFormat*	format string for ints

Returns

Error code

5.7.2.78 int timlCNNPoolingBackPropagation (timlCNNLayer ∗ *layer*)

Back propagate the gradient from the pooling layer to the previous layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.79 int timlCNNPoolingForwardPropagation (timlCNNLayer ∗ *prevLayer*)

Forward propagate form layer to layer->next.

Parameters

in	*prevLayer*	Previous layer

Returns

Error code

5.7.2.80 int timlCNNPoolingInitialize (timlCNNLayer ∗ *layer*)

Initialize the pooling layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.81 timlCNNPoolingParams timlCNNPoolingParamsDefault ( )

Return the default parameters for the pooling layer.

Returns

Default pooling layer parameters

5.7.2.82 int timlCNNPoolingReadFromTextFile (FILE * *fp1*, timlConvNeuralNetwork * *cnn*)

Read the pooling layer from a text file.

Parameters

in	*fp1*	FILE ptr to the level 1 text file
in,out	*cnn*	CNN

Returns

Error code

5.7.2.83 int timlCNNPoolingWriteToFile (FILE * *fp1*, FILE * *fp2*, FILE * *fp3*, timlCNNLayer * *layer*, timlUtilParamsLevel *level*, const char * *name*, const char * *floatFormat*, const char * *intFormat*)

Write the pooling layer to file(s)

Parameters

in,out	*fp1*	FILE ptr to the level 1 text file
in,out	*fp2*	FILE ptr to the level 2 bin file
in,out	*fp3*	FILE ptr to the level 3 bin file
in	*layer*	Layer ptr
in	*level*	Write level
in	*name*	CNN name
in	*floatFormat*	Format string for floats
in	*intFormat*	Format string for ints

Returns

Error code

5.7.2.84 int timlCNNPrint (timlConvNeuralNetwork * *cnn*)

Print out the information of the cnn.

Parameters

in,out	*cnn*	CNN

Returns

Error code

5.7.2.85 int timlCNNProfile (**timlConvNeuralNetwork** ∗ *cnn,* float ∗ *data,* int *dim,* int *num,* int ∗ *label,* int *iter*)

Profile the CNN with both timing and memory allocation.

Parameters

in	*cnn*	CNN
in	*data*	Input data batch pointer
in	*dim*	Data dimension
in	*num*	Data batch size
in	*label*	Label ptr
in	*iter*	Iterations

Returns

Error code

5.7.2.86 **timlConvNeuralNetwork**∗ timlCNNReadFromFile (const char ∗ *fileName,* int *deviceId*)

Read CNN from file(s)

Parameters

in	*fileName*	File name
in	*deviceId*	Device Id

Returns

CNN

5.7.2.87 int timlCNNReset (**timlConvNeuralNetwork** ∗ *cnn*)

Reset the parameters of the CNN.

Parameters

in,out	*cnn*	CNN

Returns

Error code

5.7.2.88 int timlCNNResetConvLayer (**timlCNNLayer** ∗ *layer*)

Reset conv layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.89 int timlCNNResetDropoutLayer (timlCNNLayer ∗ *layer*)

Reset dropout layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.90 int timlCNNResetInputLayer (timlCNNLayer ∗ *layer*)

Reset input layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.91 int timlCNNResetLinearLayer (timlCNNLayer ∗ *layer*)

Reset linear layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.92 int timlCNNResetNonlinearLayer (timlCNNLayer ∗ *layer*)

Reset nonlinear layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.93 int timlCNNResetNormLayer (timlCNNLayer ∗ *layer*)

Reset norm layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.94 int timlCNNResetPoolingLayer (timlCNNLayer ∗ *layer*)

Reset pooling layer.

Parameters

in	*layer*	Layer ptr

Returns

Error code

5.7.2.95 int timlCNNResize (timlConvNeuralNetwork ∗ *cnn,* int *row,* int *col,* int *channel*)

Resize the feature map sizes to accommodate new input feature map dimensions.

Linear layers will be converted to convolutional layer

Parameters

in	*cnn*	CNN
in	*row*	New input feature map row size
in	*col*	New input feature map col size
in	*channel*	New input feature map channel size

Returns

Error code

5.7.2.96 int timlCNNSetMode (timlConvNeuralNetwork ∗ *cnn,* timlUtilPhase *phase*)

Set the phase (train/test) of the cnn.

Parameters

in,out	*cnn*	CNN
in	*phase*	Phase

5.7.2.97 timlConvNeuralNetwork∗ timlCNNShareParams (timlConvNeuralNetwork ∗ *cnn,* int *deviceId*)

Create a new CNN that shares the parameters with the input CNN.

Unlike the clone operation, the returned CNN points to the parameters to the input CNN.

Parameters

in	*cnn*	CNN to be share parameters with
in	*deviceId*	Device Id

Returns

CNN that shares the same parameter with the input CNN

5.7.2.98 int timlCNNSupervisedTrainingWithLabelBatchMode (timlConvNeuralNetwork ∗ *cnn*, float ∗ *data*, int ∗ *label*, int *dim*, int *num*)

Supervised training with label.

Parameters

in,out	*cnn*	CNN
in	*data*	Data batch
in	*label*	Label ptr
in	*dim*	Data dimension
in	*num*	Batch size

Returns

Error code

5.7.2.99 int timlCNNSupervisedTrainingWithLabelBatchModeOpenMP (timlConvNeuralNetwork ∗ *cnn*, float ∗ *data*, int ∗ *label*, int *dim*, int *num*)

supervised training with label using openmp

Parameters

in,out	*cnn*	
in	*data*	data batch
in	*label*	
in	*dim*	data dimension
in	*num*	data number

Returns

error code

5.7.2.100 timlCNNTrainingParams timlCNNTrainingParamsDefault ( )

Return the default training parameters.

Returns

Default training parameters

5.7.2.101 int timlCNNTrainingParamsReadFromTextFile (FILE ∗ *fp*, timlConvNeuralNetwork ∗ *cnn*)

Read the training params from a text file.

Parameters

in	*fp*	FILE ptr to the level 1 text file
in,out	*cnn*	CNN

Returns

Error code

5.7.2.102 int timlCNNTrainingParamsWriteToFile (FILE ∗ *fp,* timlConvNeuralNetwork ∗ *cnn,* const char ∗ *name,* const char ∗ *floatFormat,* const char ∗ *intFormat*)

Write the training params to file(s)

Parameters

in,out	*fp*	FILE ptr to the level 1 text file
in	*cnn*	CNN
in	*name*	CNN name
in	*floatFormat*	Format string for floats
in	*intFormat*	Format string for ints

Returns

Error code

5.7.2.103 int timlCNNUpdateParams (timlConvNeuralNetwork ∗ *cnn*)

Update the parameters of the cnn.

Parameters

in,out	*cnn*	CNN

Returns

Error code

5.7.2.104 int timlCNNWriteToFile (const char ∗ *fileName,* timlConvNeuralNetwork ∗ *cnn,* timlUtilParamsLevel *level,* const char ∗ *name,* const char ∗ *floatFormat,* const char ∗ *intFormat*)

Write the cnn to file(s)

Parameters

in	*fileName*	File name
in	*cnn*	CNN
in	*level*	Parameter write level
in	*name*	Name of the cnn
in	*floatFormat*	Format string for float
in	*intFormat*	Format string for int

Returns

Error code

5.8 util

utility module

Data Structures

- struct timlUtilImage
- struct timlUtilInitializer
- struct timlUtilImageSet

Macros

- #define **TIML_UTIL_MAX_STR** 100
- #define **TIML_UTIL_PI** 3.14159265358979323846
- #define **ERROR_UTIL_OFFSET** 3000

Enumerations

- enum **timlUtilError** {
ERROR_UTIL_NULL_PTR = ERROR_UTIL_OFFSET, **ERROR_UTIL_MNIST_TRAINING_DATA_READING**, **ERROR_UTIL_MNIST_TRAINING_DATA_ALLOCATION**, **ERROR_UTIL_MNIST_TRAINING_LABEL_READING**,
ERROR_UTIL_MNIST_TRAINING_LABEL_ALLOCATION, **ERROR_UTIL_MNIST_TESTING_DATA_READING**, **ERROR_UTIL_MNIST_TESTING_DATA_ALLOCATION**, **ERROR_UTIL_MNIST_TESTING_LABEL_READING**,
ERROR_UTIL_MNIST_TESTING_LABEL_ALLOCATION, **ERROR_UTIL_CIFAR10_TRAINING_READING**, **ERROR_UTIL_CIFAR10_TRAINING_ALLOCATION**, **ERROR_UTIL_CIFAR10_TESTING_READING**,
ERROR_UTIL_CIFAR10_TESTING_ALLOCATION, **ERROR_UTIL_CIFAR100_TRAINING_READING**, **ERROR_UTIL_CIFAR100_TRAINING_ALLOCATION**, **ERROR_UTIL_CIFAR100_TESTING_READING**,
ERROR_UTIL_CIFAR100_TESTING_ALLOCATION, **ERROR_UTIL_READ_FLOAT_MATRIX**, **ERROR_UTIL_READ_INT_MATRIX**, **ERROR_UTIL_READ_FLOAT_VECTOR**,
ERROR_UTIL_READ_INT_VECTOR, **ERROR_UTIL_WRITE_FLOAT_MATRIX**, **ERROR_UTIL_WRITE_INT_MATRIX**, **ERROR_UTIL_WRITE_FLOAT_VECTOR**,
ERROR_UTIL_WRITE_INT_VECTOR, **ERROR_UTIL_MALLOC**, **ERROR_UTIL_JPEG_READING** }
- enum **timlUtilActivationType** {
Util_Sigmoid, **Util_Softmax**, **Util_Softplus**, **Util_Relu**,
Util_Nrelu, **Util_Tanh**, **Util_Linear** }
- enum **timlUtilCostFunctionType** { **Util_CrossEntropy**, **Util_MSE** }
- enum **timlUtilConvType** { **Util_Conv2D**, **Util_Corr2D** }
- enum timlUtilParamsLevel { Util_ParamsLevel1, Util_ParamsLevel2, Util_ParamsLevel3 }
- enum timlUtilAllocatorLevel { Util_AllocatorLevel1, Util_AllocatorLevel2, Util_AllocatorLevel3 }
- enum timlUtilCropType { Util_CenterCrop, Util_RandomCrop }
- enum timlUtilMirrorType { Util_Mirror, Util_NoMirror, Util_RandomMirror }
- enum **timlUtilInitializerType** { **Util_Constant**, **Util_Gaussian**, **Util_Uniform**, **Util_Xavier** }
- enum **timlUtilPhase** { **Util_Train**, **Util_Test**, **Util_Debug** }

Functions

- int timlUtilReadMNIST (const char ∗path, timlUtilImageSet ∗training, timlUtilImageSet ∗testing)

  Read MNIST database from binary files.
- int timlUtilReadCIFAR10 (const char ∗path, timlUtilImageSet ∗training, timlUtilImageSet ∗testing)

  Read CIFA10 database from binary files.
- long timlUtilDiffTime (struct timespec start, struct timespec end)

Return the time difference in micro second.

- int timlUtilRandDiscreteUniformRNG (int a, int b)

  Discrete uniform random number generator in [a, b].
- int timlUtilRandContinuousUniformRNG (float ∗x, int dim, float a, float b)

  Generate a discrete uniform random vector between (a, b)
- int timlUtilRandNormalRNG (float ∗x, int dim, float mean, float std)

  Generate a Gaussian random number.
- int timlUtilRandPerm (int ∗array, int n)

  Random permute an array.
- int timlUtilFread (void ∗ptr, size_t size, size_t nmemb, FILE ∗fp)

  Read binary file.
- int timlUtilFwrite (const void ∗ptr, size_t size, size_t nmemb, FILE ∗fp)

  Write to a binar file.
- int timlUtilConv2Valid (float ∗a, float ∗b, float ∗c, int aRow, int aCol, int bRow, int bCol)

  conv2(a, b, 'valid')
- int timlUtilConv2Full (float ∗a, float ∗b, float ∗c, int aRow, int aCol, int bRow, int bCol)

  conv2(a, b, 'full')
- int timlUtilCorr2Full (float ∗a, float ∗b, float ∗c, int aRow, int aCol, int bRow, int bCol)

  conv2(a, rot90(b,2), 'valid')
- int timlUtilConv2ImageReshapeBack (float ∗x, float ∗xReshape, int ∗index, int channel, int xDim, int indexDim, int deviceId, int threadId)

  Reshape the convolution matrix back to feature maps.
- int timlUtilConv2ImageReshapeIndex (int ∗index, int aRow, int aCol, int bRow, int bCol, int padUp, int padDown, int padLeft, int padRight, int strideX, int strideY, timlUtilConvType type)

  Create a reshaping index matrix.
- int timlUtilConv2ImageReshape (float ∗xReshape, float ∗x, int ∗index, int channel, int xDim, int indexDim, int deviceId, int threadId)

  Reshape feature maps to a format that turns 2d convolution to GEMM operation.
- timlUtilImage timlUtilReadJPEG (const char ∗name)

  read a jpg image
- int timlUtilReadFixedSizeJPEG (const char ∗name, float ∗data, int row, int col, int channel)

  Read a jpg image with known size information to avoid frequent allocation and deallocation of data.
- char ∗∗ timlUtilScanJPEG (const char ∗dirName, int ∗imageNum)

  Return an array of jpg image names in the directory.
- void timlUtilBLASdgemm (const enum CBLAS_TRANSPOSE TransA, const enum CBLAS_TRANSPOSE TransB, const int M, const int N, const int K, const double alpha, const double ∗A, const double ∗B, const double beta, double ∗C, int deviceId, int threadId)

  Double general matrix matrix multiplication C = alpha ∗ op(A) ∗ op(B) + beta ∗ C op(A) : M∗K op(B) : K∗N.
- void timlUtilBLASsgemm (const enum CBLAS_TRANSPOSE TransA, const enum CBLAS_TRANSPOSE TransB, const int M, const int N, const int K, const float alpha, const float ∗A, const float ∗B, const float beta, float ∗C, int deviceId, int threadId)

  Float general matrix matrix multiplication C = alpha ∗ op(A) ∗ op(B) + beta ∗ C op(A) : M∗K op(B) : K∗N.
- void timlUtilBLASdgemv (const enum CBLAS_TRANSPOSE TransA, const int M, const int N, const double alpha, const double ∗A, const double ∗x, const double beta, double ∗y, int deviceId, int threadId)

  Double general matrix vector multiplication y = alpha ∗ op(A) ∗ x + beta ∗ y op(A): M∗N.
- void timlUtilBLASsgemv (const enum CBLAS_TRANSPOSE TransA, const int M, const int N, const float alpha, const float ∗A, const float ∗x, const float beta, float ∗y, int deviceId, int threadId)

  Float general matrix vector multiplication y = alpha ∗ op(A) ∗ x + beta ∗ y op(A): M∗N.
- void timlUtilBLASsaxpy (const int N, const float alpha, const float ∗X, float ∗Y, int deviceId, int threadId)

  Float vector addition Y = alpha ∗ X + Y.
- void timlUtilBLASdaxpy (const int N, const double alpha, const double ∗X, double ∗Y, int deviceId, int threadId)

  Double vector addition Y = alpha ∗ X + Y.

- void timlUtilBLASscopy (const int N, const float ∗X, float ∗Y, int deviceId, int threadId)

  Float vector copy Y = X.
- void timlUtilBLASdcopy (const int N, const double ∗X, double ∗Y, int deviceId, int threadId)

  Double vector copy Y = X.
- void timlUtilBLASsger (const int M, const int N, const float alpha, float ∗x, float ∗y, float ∗A, int deviceId, int threadId)

  Float vector outer product A = alpha∗x∗y' + A; x: M y: N.
- void timlUtilBLASdger (const int M, const int N, const double alpha, double ∗x, double ∗y, double ∗A, int deviceId, int threadId)

  Double vector outer product A = alpha∗x∗y' + A; x: M y: N.
- void timlUtilBLASdscal (const int N, const double alpha, double ∗X, int deviceId, int threadId)

  Double vector scaling x = alpha ∗ x.
- void timlUtilBLASsscal (const int N, const float alpha, float ∗X, int deviceId, int threadId)

  Float vector scaling x = alpha ∗ x.
- int timlUtilVectorResetFloat (float ∗a, int m, float val, int deviceId, int threadId)

  Reset a float vector.
- int timlUtilVectorResetInt (int ∗a, int m, int val, int deviceId, int threadId)

  Reset an int vector.
- float timlUtilVectorSumFloat (float ∗a, int n)

  Calculate the sum of a float vector.
- int timlUtilVectorSortFloat (float ∗a, int n)

  Sort an array in descending order.
- int timlUtilVectorSortIndexFloat (float ∗a, int ∗index, int n)

  Sort an array in descending order and return the indices of the original elements in the sorted array.
- float timlUtilVectorMaxFloat (float ∗x, int n, int inc)

  Return the max value in the array.
- int timlUtilVectorMaxIndexFloat (float ∗x, int n, int inc)

  Return the max value index in the array.
- int timlUtilElementWiseMultiply (float ∗a, const float ∗b, const float ∗c, int dim, int deviceId, int threadId)

  Element wise multiply c = a.∗b.
- int timlUtilSubtract (float ∗x, float y, int deviceId, int threadId)

  Subtract operation.
- int timlUtilSigmoid (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Sigmoid.
- int timlUtilSigmoidDerivative (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Sigmoid derivative.
- int timlUtilRelu (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Rectified linear unit.
- int timlUtilReluDerivative (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Rectified linear unit derivative.
- int timlUtilTanhDerivative (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Tanh derivative.
- float timlUtilMultinomialCrossEntropy (float ∗x, int label, int n)

  Calculate the mutlinomial cross entropy between x and label.
- float timlUtilMeanSqaureError (float ∗x, int label, int n)

  Calculate the mean square error between x and label.
- int timlUtilSoftmax (float ∗x, float ∗y, int row, int col, int channel, int deviceId, int threadId)

  Softmax function.
- int timlUtilClassifyAccuracy (int ∗label, int topN, int num, int ∗trueLabel)

  Calculate the classification accuracy.

- void timlUtilTransform (float ∗dataOut, float ∗dataIn, float ∗dataHost, int channel, int row, int col, int rowOffset, int colOffset, int rowIn, int colIn, float scale, float ∗mean, timlUtilMirrorType mirrorType, int deviceId, int threadId)

  Transform the raw input data with preprocessing.
- int timlUtilMaxPooling (float ∗outputMap, int ∗maxIndex, float ∗inputMap, int row, int col, int channel, int prevRow, int prevCol, int scaleRow, int scaleCol, int padUp, int padLeft, int strideX, int strideY, timlUtilPhase phase, int deviceId, int threadId)

  Max pooling.
- int timlUtilUndoMaxPooling (float ∗prevDelta, int ∗maxIndex, float ∗delta, int dim, int deviceId, int threadId)

  Undo max pooling.
- int timlUtilMeanPooling (float ∗outputMap, float ∗inputMap, int row, int col, int channel, int prevRow, int prevCol, int scaleRow, int scaleCol, int padUp, int padLeft, int strideX, int strideY, int deviceId, int threadId)

  Mean pooling.
- int timlUtilUndoMeanPooling (float ∗prevDelta, float ∗delta, int row, int col, int channel, int prevRow, int prevCol, int scaleRow, int scaleCol, int padUp, int padLeft, int strideX, int strideY, int deviceId, int threadId)

  Undo mean pooling.
- int timlUtilLocalContrastNormalize (float ∗inputMap, float ∗outputMap, float ∗denom, int row, int col, int channel, int N, float alpha, float beta, int deviceId, int threadId)

  Local contrast normalization.
- int timlUtilLocalContrastUnnormalize (float ∗prevDelta, float ∗prevFeatureMap, float ∗delta, float ∗featureMap, float ∗denom, int row, int col, int channel, int N, float alpha, float beta, int deviceId, int threadId)

  Local contrast unnormalization.
- int timlUtilMasking (float ∗inputMap, float ∗outputMap, int ∗mask, unsigned int ∗randomVector, int dim, float prob, int deviceId, int threadId)

  Masking feature maps.
- int timlUtilUnmasking (float ∗inputDelta, float ∗outputDelta, int ∗mask, int dim, float prob, int deviceId, int threadId)

  Masking feature maps.
- int timlUtilTanh (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Tanh.
- int timlUtilMalloc (void ∗∗devPtr, size_t size)

  memory allocation
- void timlUtilFree (void ∗ptr)

  Free pointer.
- uint32_t timlUtilReverseEndian32 (register uint32_t i)

  Reverse the 32 bit endian pattern.
- int timlUtilElementWiseFunction (float ∗x, float ∗y, int n, float(∗func)(float))

  Apply a function on each element of the array.

5.8.1 Detailed Description

utility module

5.8.2 Enumeration Type Documentation

5.8.2.1 enum timlUtilAllocatorLevel

Enumerator

Util_AllocatorLevel1 training mode

Util_AllocatorLevel2 testing mode

Util_AllocatorLevel3 testing mode with memory pool

5.8.2.2 enum timlUtilCropType

Enumerator

Util_CenterCrop crop the picture at the center

Util_RandomCrop randomly crop the picture

5.8.2.3 enum timlUtilMirrorType

Enumerator

Util_Mirror mirror the picture

Util_NoMirror do not mirror the picture

Util_RandomMirror randomly mirror the picture according to Bernoulli(0,1)

5.8.2.4 enum timlUtilParamsLevel

Enumerator

Util_ParamsLevel1 structure text file only

Util_ParamsLevel2 structure text + parameter binary

Util_ParamsLevel3 structure text + parameter binary + state binary

5.8.3 Function Documentation

5.8.3.1 int timlUtilClassifyAccuracy (int ∗ *label,* int *topN,* int *num,* int ∗ *trueLabel*)

Calculate the classification accuracy.

Parameters

`in`	*label*	Label matrix, size = num∗topN
`in`	*topN*	Top N labels
`in`	*num*	Number of samples
`in`	*trueLabel*	True label array, size = num

Returns

Total number of correct labels

5.8.3.2 int timlUtilConv2Full (float ∗ *a,* float ∗ *b,* float ∗ *c,* int *aRow,* int *aCol,* int *bRow,* int *bCol*)

conv2(a, b, ’full’)

Parameters

`in`	*a*	
`in`	*b*	
`out`	*c*	c = conv2(a, b, ’full’)
`in`	*aRow*	a row size

in	*aCol*	c col size
in	*bRow*	b row size
in	*bCol*	b col size

Returns

Error code

5.8.3.3 int timlUtilConv2ImageReshape (float ∗ *xReshape,* float ∗ *x,* int ∗ *index,* int *channel,* int *xDim,* int *indexDim,* int *deviceId,* int *threadId*)

Reshape feature maps to a format that turns 2d convolution to GEMM operation.

Parameters

out	*xReshape*	Reshaped feature map
in	*x*	Feature map
in	*index*	Reshaping index matrix
in	*channel*	The number of channels in the feature maps
in	*xDim*	Dimension of the feature map (row∗col)
in	*indexDim*	Dimension of the index matrix
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.4 int timlUtilConv2ImageReshapeBack (float ∗ *x,* float ∗ *xReshape,* int ∗ *index,* int *channel,* int *xDim,* int *indexDim,* int *deviceId,* int *threadId*)

Reshape the convolution matrix back to feature maps.

Parameters

in	*x*	feature map
out	*xReshape*	Reshaped feature map
in	*index*	Reshaping index matrix
in	*channel*	The number of channels in the feature map
in	*xDim*	Dimension of the feature map (row∗col)
in	*indexDim*	Dimension of the index matrix
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.5 int timlUtilConv2ImageReshapeIndex (int ∗ *index,* int *aRow,* int *aCol,* int *bRow,* int *bCol,* int *padUp,* int *padDown,* int *padLeft,* int *padRight,* int *strideX,* int *strideY,* timlUtilConvType *type*)

Create a reshaping index matrix.

Feature maps need to be reshaped so that a 2d convolution can be converted to a GEMM operation. The reshaping index matrix records the index mapping between the original feature maps and the reshaped feature maps

Parameters

out	*index*	Reshaping index matrix
in	*aRow*	Feature map row
in	*aCol*	Feature map col
in	*bRow*	Kernel row
in	*bCol*	Kernel col
in	*padUp*	Padding for the up border of the image
in	*padDown*	Padding for the down border of the image
in	*padLeft*	Padding for the left border of the image
in	*padRight*	Padding for the right border of the image
in	*strideX*	Horizontal stride for the kernel
in	*strideY*	Vertical stride for the kernel
in	*type*	Convolution or correlation

Returns

Error code

5.8.3.6 int timlUtilConv2Valid (float $*$ *a,* float $*$ *b,* float $*$ *c,* int *aRow,* int *aCol,* int *bRow,* int *bCol*)

conv2(a, b, 'valid')

Parameters

in	*a*	a matrix
in	*b*	b matrix
out	*c*	c = conv2(a, b, 'valid')
in	*aRow*	a row size
in	*aCol*	a col size
in	*bRow*	b row size
in	*bCol*	b col size

Returns

Error code

5.8.3.7 int timlUtilCorr2Full (float $*$ *a,* float $*$ *b,* float $*$ *c,* int *aRow,* int *aCol,* int *bRow,* int *bCol*)

conv2(a, rot90(b,2), 'valid')

Parameters

in	*a*	a matrix
in	*b*	b matrix
out	*c*	c = conv2(a, rot90(b,2), 'valid')
in	*aRow*	a row size
in	*aCol*	a col size
in	*bRow*	b row size
in	*bCol*	b col size

Returns

Error code

5.8.3.8 long timlUtilDiffTime (struct timespec *start,* struct timespec *end*)

Return the time difference in micro second.

Parameters

in	*start*	Start time
in	*end*	End time return Time difference

5.8.3.9 int timlUtilElementWiseFunction (float ∗ *x*, float ∗ *y*, int *n*, float(∗)(float) *func*)

Apply a function on each element of the array.

Parameters

in	*x*	Input array
out	*y*	Output array
in	*n*	Array size
in	*func*	Function pointer

Returns

Error code

5.8.3.10 int timlUtilElementWiseMultiply (float ∗ *a*, const float ∗ *b*, const float ∗ *c*, int *dim*, int *deviceId*, int *threadId*)

Element wise multiply c = a.∗b.

Parameters

in	*a*	a vector
in	*b*	b vector
out	*c*	c = a.∗b
in	*dim*	Dimension of a,b,c
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.11 int timlUtilFread (void ∗ *ptr*, size_t *size*, size_t *nmemb*, FILE ∗ *fp*)

Read binary file.

Parameters

out	*ptr*	Pointer
in	*size*	Size of array
in	*nmemb*	Array element size
in	*fp*	File pointer

Returns

The number of successfully read elements

5.8.3.12 void timlUtilFree (void ∗ *ptr*)

Free pointer.

Parameters

in	*ptr*	Memory pointer

5.8.3.13 int timlUtilFwrite (const void ∗ *ptr,* size_t *size,* size_t *nmemb,* FILE ∗ *fp*)

Write to a binar file.

Parameters

out	*ptr*	Memory pointer
in	*size*	Array size
in	*nmemb*	Array element size
in	*fp*	File pointer

Returns

Number of successfully written elements

5.8.3.14 int timlUtilLocalContrastNormalize (float ∗ *inputMap,* float ∗ *outputMap,* float ∗ *denom,* int *row,* int *col,* int *channel,* int *N,* float *alpha,* float *beta,* int *deviceId,* int *threadId*)

Local contrast normalization.

Parameters

in	*inputMap*	Input feature map
out	*outputMap*	Output feature map
out	*denom*	Feature map denom
in	*row*	Input feature map row
in	*col*	Input feature map col
in	*channel*	Input feature map channel
in	*N*	Channel span
in	*alpha*	Alpha
in	*beta*	Beta
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.15 int timlUtilLocalContrastUnnormalize (float ∗ *prevDelta,* float ∗ *prevFeatureMap,* float ∗ *delta,* float ∗ *featureMap,* float ∗ *denom,* int *row,* int *col,* int *channel,* int *N,* float *alpha,* float *beta,* int *deviceId,* int *threadId*)

Local contrast unnormalization.

Parameters

out	*prevDelta*	Previous delta
in	*prevFeatureMap*	Previous feature map
in	*delta*	Delta

in	*featureMap*	Feature map
in	*denom*	Feature map denom
in	*row*	Feature map row
in	*col*	Feature map col
in	*channel*	Feature map channel
in	*N*	Channel span
in	*alpha*	Alpha
in	*beta*	Beta
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.16 int timlUtilMalloc (void ∗∗ *devPtr,* size_t *size*)

memory allocation

Parameters

in,out	*devPtr*	device ptr
in	*size*	allocation size in byte

Returns

error code

5.8.3.17 int timlUtilMasking (float ∗ *inputMap,* float ∗ *outputMap,* int ∗ *mask,* unsigned int ∗ *randomVector,* int *dim,* float *prob,* int *deviceId,* int *threadId*)

Masking feature maps.

Parameters

in	*inputMap*	Input feature map
out	*outputMap*	Output feature map
out	*mask*	Mask vector of values {0,1}
in	*randomVector*	A uniform random vector in [0,1]
in	*dim*	Dimension of the feature map (row∗col∗channel)
in	*prob*	Dropout probability
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.18 int timlUtilMaxPooling (float ∗ *outputMap,* int ∗ *maxIndex,* float ∗ *inputMap,* int *row,* int *col,* int *channel,* int *prevRow,* int *prevCol,* int *scaleRow,* int *scaleCol,* int *padUp,* int *padLeft,* int *strideX,* int *strideY,* timlUtilPhase *phase,* int *deviceId,* int *threadId*)

Max pooling.

Parameters

out	*outputMap*	Output feature map
in	*maxIndex*	Max value index map
in	*inputMap*	Input feature map
in	*row*	Output feature map row
in	*col*	Output feature map col
in	*channel*	Output feature map channel
in	*prevRow*	Previous feature map row
in	*prevCol*	Previous feature map col
in	*scaleRow*	Scaling window row size
in	*scaleCol*	Scaling window col size
in	*padUp*	Upper border padding for the input feature map
in	*padLeft*	Left border padding for the input feature map
in	*strideX*	Window stride in x direction
in	*strideY*	Window stride in y direction
in	*phase*	CNN phase
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.19 int timlUtilMeanPooling (float ∗ *outputMap,* float ∗ *inputMap,* int *row,* int *col,* int *channel,* int *prevRow,* int *prevCol,* int *scaleRow,* int *scaleCol,* int *padUp,* int *padLeft,* int *strideX,* int *strideY,* int *deviceId,* int *threadId*)

Mean pooling.

Parameters

out	*outputMap*	Output feature map
in	*inputMap*	Input feature map
in	*row*	Output feature map row
in	*col*	Output feature map col
in	*channel*	Output feature map channel
in	*prevRow*	Previous feature map row
in	*prevCol*	Previous feature map col
in	*scaleRow*	Scaling window row size
in	*scaleCol*	Scaling window col size
in	*padUp*	Upper border padding for the input feature map
in	*padLeft*	Left border padding for the input feature map
in	*strideX*	Window stride in x direction
in	*strideY*	Window stride in y direction
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.20 float timlUtilMeanSqaureError (float ∗ *x,* int *label,* int *n*)

Calculate the mean square error between x and label.

Parameters

in	*x*	Input data
in	*label*	Label
in	*n*	Data size

Returns

Mean square error

5.8.3.21 float timlUtilMultinomialCrossEntropy (float ∗ *x*, int *label*, int *n*)

Calculate the mutlinomial cross entropy between x and label.

Parameters

in	*x*	Input data
in	*label*	Label
in	*n*	Data size

Returns

Cross entropy

5.8.3.22 int timlUtilRandContinuousUniformRNG (float ∗ *x*, int *dim*, float *a*, float *b*)

Generate a discrete uniform random vector between (a, b)

Parameters

in	*x*	Generated random vector
in	*dim*	Dimension
in	*a*	Lower bound
in	*b*	Upper bound

Returns

Random vector

5.8.3.23 int timlUtilRandDiscreteUniformRNG (int *a*, int *b*)

Discrete uniform random number generator in [a, b].

Parameters

in	*a*	Lower bound
in	*b*	Upper bound

Returns

Random integer

5.8.3.24 int timlUtilRandNormalRNG (float ∗ *x*, int *dim*, float *mean*, float *std*)

Generate a Gaussian random number.

Parameters

out	*x*	Guassian random vector
in	*dim*	Vector dimension
in	*mean*	Mean
in	*std*	Standard deviation

Returns

Error code

5.8.3.25 int timlUtilRandPerm (int ∗ *array,* int *n*)

Random permute an array.

Parameters

in,out	*array*	Array
in	*n*	Array size

Returns

Error code

5.8.3.26 int timlUtilReadCIFAR10 (const char ∗ *path,* timlUtilImageSet ∗ *training,* timlUtilImageSet ∗ *testing*)

Read CIFA10 database from binary files.

Parameters

out	*training*	Training database
out	*testing*	Testing database

Returns

Error code

5.8.3.27 int timlUtilReadFixedSizeJPEG (const char ∗ *name,* float ∗ *data,* int *row,* int *col,* int *channel*)

Read a jpg image with known size information to avoid frequent allocation and deallocation of data.

Parameters

in	*name*	Image name
out	*data*	Data
in	*row*	Row
in	*col*	Col
in	*channel*	Channel

Returns

Error code

5.8.3.28 timlUtilImage timlUtilReadJPEG (const char ∗ *name*)

read a jpg image

Parameters

in	*name*	image name

Returns

timlUtilImage structure

5.8.3.29 int timlUtilReadMNIST (const char ∗ *path,* timlUtilImageSet ∗ *training,* timlUtilImageSet ∗ *testing*)

Read MNIST database from binary files.

Parameters

in	*path*	Database path
out	*training*	Training database
out	*testing*	Testing database

Returns

Error code

5.8.3.30 int timlUtilRelu (float ∗ *x,* float ∗ *y,* int *n,* int *deviceId,* int *threadId*)

Rectified linear unit.

Parameters

in	*x*	Input
out	*y*	Outupt
in	*n*	Input size
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.31 int timlUtilReluDerivative (float ∗ *x,* float ∗ *y,* int *n,* int *deviceId,* int *threadId*)

Rectified linear unit derivative.

Parameters

in	*x*	Input
out	*y*	Derivative of relu(x)
in	*n*	Input size
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.32 uint32_t timlUtilReverseEndian32 (register uint32_t *i*)

Reverse the 32 bit endian pattern.

Parameters

in	*i*	Integer input

Returns

Integer output

5.8.3.33 char ∗∗ timlUtilScanJPEG (const char ∗ *dirName,* int ∗ *imageNum*)

Return an array of jpg image names in the directory.

Parameters

in	*dirName*	Directory name
out	*imageNum*	Image number

Returns

Image name array

5.8.3.34 int timlUtilSigmoid (float ∗ *x,* float ∗ *y,* int *n,* int *deviceId,* int *threadId*)

Sigmoid.

Parameters

in	*x*	Input
out	*y*	Output y = sigmoid(x)
in	*n*	Input size
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.35 int timlUtilSigmoidDerivative (float ∗ *x,* float ∗ *y,* int *n,* int *deviceId,* int *threadId*)

Sigmoid derivative.

Parameters

in	*x*	Input
out	*y*	Outupt y = derivative of sigmoid(x)
in	*n*	Input size
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.36 int timlUtilSoftmax (float ∗ *x,* float ∗ *y,* int *row,* int *col,* int *channel,* int *deviceId,* int *threadId*)

Softmax function.

Parameters

in	*x*	Input
out	*y*	Outupt
in	*row*	x row size
in	*col*	x col size
in	*channel*	x channel size
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.37 int timlUtilSubtract (float ∗ *x,* float *y,* int *deviceId,* int *threadId*)

Subtract operation.

Parameters

in,out	*x*	x = x - y
in	*y*	Subtract constant
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.38 int timlUtilTanh (float ∗ *x,* float ∗ *y,* int *n,* int *deviceId,* int *threadId*)

Tanh.

Parameters

in	*x*	Input
out	*y*	Output = tanh(x)
in	*n*	Input size
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.39 int timlUtilTanhDerivative (float ∗ *x,* float ∗ *y,* int *n,* int *deviceId,* int *threadId*)

Tanh derivative.

Parameters

in	*x*	Input

out	*y*	Output = derivative of tanh(x)
in	*n*	Input size
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.40 void timlUtilTransform (float ∗ *dataOut,* float ∗ *dataIn,* float ∗ *dataHost,* int *channel,* int *row,* int *col,* int *rowOffset,* int *colOffset,* int *rowIn,* int *colIn,* float *scale,* float ∗ *mean,* timlUtilMirrorType *mirrorType,* int *deviceId,* int *threadId*)

Transform the raw input data with preprocessing.

Parameters

out	*dataOut*	Output data, i.e. the input feature map
out	*dataIn*	A copy of the input data
in	*dataHost*	Input data
in	*channel*	Input feature map channel
in	*row*	Input feature map row
in	*col*	Input feature map col
in	*rowOffset*	Row offset with regard to the raw input data
in	*colOffset*	Col offset with regard to the raw input data
in	*rowIn*	Raw input data row
in	*colIn*	Raw input data col
in	*scale*	Scaling factor
in	*mean*	Input data mean
in	*mirrorType*	Whether to mirror the raw input data
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.41 int timlUtilUndoMaxPooling (float ∗ *prevDelta,* int ∗ *maxIndex,* float ∗ *delta,* int *dim,* int *deviceId,* int *threadId*)

Undo max pooling.

Parameters

out	*prevDelta*	Previous layer delta
in	*maxIndex*	Max feature map value indices of the current layer
in	*delta*	Current layer delta
in	*dim*	Dimension of the current layer feature map
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.42 int timlUtilUndoMeanPooling (float ∗ *prevDelta,* float ∗ *delta,* int *row,* int *col,* int *channel,* int *prevRow,* int *prevCol,* int *scaleRow,* int *scaleCol,* int *padUp,* int *padLeft,* int *strideX,* int *strideY,* int *deviceId,* int *threadId*)

Undo mean pooling.

Parameters

out	*prevDelta*	Previous layer delta
in	*delta*	Current feature map delta
in	*row*	Current feature map row
in	*col*	Current feature map col
in	*channel*	Current feature map channel
in	*prevRow*	Previous feature map row
in	*prevCol*	Previous feature map col
in	*scaleRow*	Scaling window row size
in	*scaleCol*	Scaling window col size
in	*padUp*	Upper border padding for the previous feature map
in	*padLeft*	Left border padding for the previous feature map
in	*strideX*	Window stride in x direction
in	*strideY*	Window stride in y direction
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.43 int timlUtilUnmasking (float ∗ *inputDelta,* float ∗ *outputDelta,* int ∗ *mask,* int *dim,* float *prob,* int *deviceId,* int *threadId*)

Masking feature maps.

Parameters

in	*inputDelta*	Current feature map delta
out	*outputDelta*	Previous feature map delta
in	*mask*	Masking vector
in	*dim*	Dimension of the masking vector
in	*prob*	Dropout probability
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.44 float timlUtilVectorMaxFloat (float ∗ *x,* int *n,* int *inc*)

Return the max value in the array.

Parameters

in	*x*	Input array
in	*n*	Array size
in	*inc*	Increment

Returns

Max value

5.8.3.45 int timlUtilVectorMaxIndexFloat (float ∗ *x,* int *n,* int *inc*)

Return the max value index in the array.

Parameters

in	*x*	Input array
in	*n*	Array size
in	*inc*	Increment

Returns

Max value index

5.8.3.46 int timlUtilVectorResetFloat (float ∗ *a,* int *m,* float *val,* int *deviceId,* int *threadId*)

Reset a float vector.

Parameters

in,out	*a*	Vector
in	*m*	Vector size
in	*val*	Value
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.47 int timlUtilVectorResetInt (int ∗ *a,* int *m,* int *val,* int *deviceId,* int *threadId*)

Reset an int vector.

Parameters

in,out	*a*	Vector
in	*m*	Vector size
in	*val*	Value
in	*deviceId*	Device id
in	*threadId*	Thread id

Returns

Error code

5.8.3.48 int timlUtilVectorSortFloat (float ∗ *a,* int *n*)

Sort an array in descending order.

Parameters

in,out	*a*	Array
in	*n*	Array size

Returns

Error code

5.8.3.49 int timlUtilVectorSortIndexFloat (float ∗ *a,* int ∗ *index,* int *n*)

Sort an array in descending order and return the indices of the original elements in the sorted array.

Parameters

in	*a*	Array
out	*index*	Sorted index
in	*n*	Array size

Returns

Error code

5.8.3.50 float timlUtilVectorSumFloat (float ∗ *a,* int *n*)

Calculate the sum of a float vector.

Parameters

in,out	*a*	Vector
in	*n*	Vector size

Returns

Sum

5.9 testCNN

Test CNN module.

Functions

- int testCNNSimpleTraining ()

  Simple training function test.
- int testCNNSimpleClone ()

  Simple clone function test.
- int testCNNSimpleIO ()

  Simple read/write function test.
- int testCNNSimpleResize ()

  Simple resize function test.
- int testCNNSimpleProfile ()

  simple profile function test
- int testCNNSimpleShare ()

  Simple share function test.

5.9.1 Detailed Description

Test CNN module.

5.9.2 Function Documentation

5.9.2.1 int testCNNSimpleClone ( )

Simple clone function test.

Returns

Error code

5.9.2.2 int testCNNSimpleIO ( )

Simple read/write function test.

Returns

Error code

5.9.2.3 int testCNNSimpleProfile ( )

simple profile function test

Returns

error code

5.9.2.4 int testCNNSimpleResize ( )

Simple resize function test.

Returns

Error code

5.9.2.5 int testCNNSimpleShare ( )

Simple share function test.

Returns

Error code

5.9.2.6 int testCNNSimpleTraining ( )

Simple training function test.

Returns

Error code

5.10 testUtil

Test utility module.

Functions

- int testUtilBLAS (void)

    BLAS function test.
- int testUtilConv2 (void)

    2d convoultion function test
- int testUtilSort ()

    Sort function test.

5.10.1 Detailed Description

Test utility module.

5.10.2 Function Documentation

5.10.2.1 int testUtilBLAS (void)

BLAS function test.

Returns

Error code

5.10.2.2 int testUtilConv2 (void)

2d convoultion function test

Returns

Error code

5.10.2.3 int testUtilSort ( )

Sort function test.

Returns

Error code

5.11 app

Applications.

Modules

- appCNN

  CNN applications.

5.11.1 Detailed Description

Applications.

5.12 appCNN

CNN applications.

Modules

- appCNNClass

  CNN classification application.
- appCNNConvertImageNet

  ImageNet 2012 database conversion applications.
- appCNNConvertSBD

  Stanford background dataset conversion applications.
- appCNNInteropCaffe

  CNN Caffe interoperation applications.
- appCNNScene

  CNN scene labeling application.

5.12.1 Detailed Description

CNN applications.

5.13 benchmark

Bechmarks.

Modules

- benchmarkCNN

  CNN benchmarks.

5.13.1 Detailed Description

Bechmarks.

5.14 benchmarkCNN

CNN benchmarks.

Modules

- benchmarkCNNClass

  Benchmark CNN classification.

5.14.1 Detailed Description

CNN benchmarks.

5.15 test

Test.

Modules

- testCNN

  Test CNN module.
- testUtil

  Test utility module.

Functions

- int testCNNSimpleProfile ()

  simple profile function test

5.15.1 Detailed Description

Test.

5.15.2 Function Documentation

5.15.2.1 int testCNNSimpleProfile ( )

simple profile function test

Returns

error code

Chapter 6

Data Structure Documentation

6.1 _timlCNNLayer_ Struct Reference

Data Fields

- int **id**
- timlCNNLayerType **type**
- int **row**
- int **col**
- int **channel**
- float ∗ **featureMap**
- float ∗ delta
- timlUtilPhase **phase**
- timlUtilAllocatorLevel **allocatorLevel**
- timlCNNDropoutParams dropoutParams
- timlCNNInputParams **inputParams**
- timlCNNConvParams **convParams**
- timlCNNNormParams **normParams**
- timlCNNPoolingParams **poolingParams**
- timlCNNNonlinearParams **nonlinearParams**
- timlCNNLinearParams **linearParams**
- struct _timlCNNLayer_ ∗ prev
- struct _timlCNNLayer_ ∗ **next**
- struct _timlConvNeuralNetwork_ ∗ **cnn**

6.1.1 Field Documentation

6.1.1.1 float∗ _timlCNNLayer_::delta

partial derivative of the cost function with regard to each kernel

6.1.1.2 timlCNNDropoutParams _timlCNNLayer_::dropoutParams

only one of the layer-specific params structure is valid

6.1.1.3 struct _timlCNNLayer_ ∗ _timlCNNLayer_::prev

layers are connected with double linked list

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

- timlCNN.h

6.2 _timlConvNeuralNetwork_ Struct Reference

Data Fields

- float ∗ memPool
- int memPoolSize
- int **deviceId**
- int **threadId**
- timlCNNLayer ∗ **head**
- timlCNNLayer ∗ **tail**
- timlCNNTrainingParams **params**

6.2.1 Field Documentation

6.2.1.1 float∗ _timlConvNeuralNetwork_::memPool

used by allocatorLevel3 mode to store the feature maps

6.2.1.2 int _timlConvNeuralNetwork_::memPoolSize

size of the memory pool

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

- timlCNN.h

6.3 appCNNSceneDataSet Struct Reference

Data Fields

- int **num**
- int **row**
- int **col**
- int **channel**
- const char ∗ **imageFileNameStr**
- const char ∗ **labelFileNameStr**
- int patchSize

6.3.1 Field Documentation

6.3.1.1 int appCNNSceneDataSet::patchSize

image patch(square) size

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

- appCNNScene.h

6.4 timlCNNConvParams Struct Reference

Data Fields

- int **inputFeatureMapChannel**
- int **outputFeatureMapChannel**
- int **kernelRow**
- int **kernelCol**
- int **padUp**
- int **padDown**
- int **padLeft**
- int **padRight**
- int **strideX**
- int **strideY**
- timlUtilConvType **type**
- float ∗ prevFeatureMapReshape
- int ∗ prevFeatureMapReshapeIndex
- float ∗ **kernelGradAccum**
- float ∗ kernel
- float ∗ **kernelInc**
- float **kernelDecayFactor**
- float **kernelLearningFactor**
- timlUtilInitializer **kernelInit**
- int ∗ connectivity
- float ∗ **bias**
- float ∗ **biasGradAccum**
- float ∗ **biasInc**
- float ∗ **biasMultiplier**
- float **biasLearningFactor**
- timlUtilInitializer **biasInit**
- bool **shared**

6.4.1 Field Documentation

6.4.1.1 int∗ timlCNNConvParams::connectivity

connectivity matrix (if prev->featureMap(i) is connected to layer->featureMap(j) by a kernel)

6.4.1.2 float∗ timlCNNConvParams::kernel

size = (channel) ∗ (kernelRow∗kernelCol∗prev->channel)

6.4.1.3 float∗ timlCNNConvParams::prevFeatureMapReshape

reshape the feature map of the previous layer to size (prev->channel∗kernelRow∗kernelCol) ∗ (row∗col)

6.4.1.4 int∗ timlCNNConvParams::prevFeatureMapReshapeIndex

the reshape matrix of size (kernelRow∗kernelCol) ∗ (row∗col)

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

- timlCNN.h

6.5 timlCNNDataSet Struct Reference

Data Fields

- int **num**
- int **channel**
- int **row**
- int **col**
- float ∗ data
- int ∗ label

6.5.1 Field Documentation

6.5.1.1 float∗ timlCNNDataSet::data

size = (rowSize ∗ colSize ∗ channel) ∗ num

6.5.1.2 int∗ timlCNNDataSet::label

size = num

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

- timlCNN.h

6.6 timlCNNDropoutParams Struct Reference

Data Fields

- int ∗ mask
- unsigned int ∗ randomVector
- float prob

6.6.1 Field Documentation

6.6.1.1 int∗ timlCNNDropoutParams::mask

a mask matrix of values (0,1)

6.6.1.2 float timlCNNDropoutParams::prob

dropout probability

6.6.1.3 unsigned int∗ timlCNNDropoutParams::randomVector

dropout random unsigned int vector

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

- timlCNN.h

6.7 timlCNNInputParams Struct Reference

Data Fields

- int row
- int col
- int channel
- int ∗ channelPermute
- float ∗ mean
- float **scale**
- float ∗ inputData
- timlUtilCropType **trainingCropType**
- timlUtilMirrorType **trainingMirrorType**
- timlUtilCropType **testingCropType**
- timlUtilMirrorType **testingMirrorType**
- bool shared

6.7.1 Field Documentation

6.7.1.1 int timlCNNInputParams::channel

raw data channel size

6.7.1.2 int∗ timlCNNInputParams::channelPermute

channel permutation order

6.7.1.3 int timlCNNInputParams::col

raw data col size

6.7.1.4 float∗ timlCNNInputParams::inputData

raw data

6.7.1.5 float∗ timlCNNInputParams::mean

mean of the raw data

6.7.1.6 int timlCNNInputParams::row

raw data row size

6.7.1.7 bool timlCNNInputParams::shared

if this layer shares the same mean with some other layer

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

- timlCNN.h

6.8 timlCNNLinearParams Struct Reference

Data Fields

- int dim
- int prevDim
- float ∗ **weight**
- float ∗ **weightInc**
- float ∗ **weightGradAccum**
- float ∗ **bias**
- float ∗ **biasInc**
- float ∗ **biasGradAccum**
- float **weightDecayFactor**
- timlUtilInitializer **weightInit**
- timlUtilInitializer **biasInit**
- float **weightLearningFactor**
- float **biasLearningFactor**
- bool shared

6.8.1 Field Documentation

6.8.1.1 int timlCNNLinearParams::dim

1d dimension of the layer

6.8.1.2 int timlCNNLinearParams::prevDim

1d dimension of the previous layer

6.8.1.3 bool timlCNNLinearParams::shared

if this layer shares the parameters from other layer

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

- timlCNN.h

6.9 timlCNNNonlinearParams Struct Reference

Data Fields

- timlUtilActivationType **type**
- float ∗ **derivative**

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

- timlCNN.h

6.10 timlCNNNormParams Struct Reference

Data Fields

- timlCNNNormType **type**
- int **N**
- float **alpha**
- float **beta**
- float ∗ denom

6.10.1 Field Documentation

6.10.1.1 float∗ timlCNNNormParams::denom

denominator

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

- timlCNN.h

6.11 timlCNNPoolingParams Struct Reference

Data Fields

- timlCNNPoolingType **type**
- int scaleRow
- int scaleCol
- int **padUp**
- int **padDown**
- int **padLeft**
- int **padRight**
- int strideX
- int strideY
- int ∗ maxIndex

6.11.1 Field Documentation

6.11.1.1 int∗ timlCNNPoolingParams::maxIndex

recode the indices of the max pooling value so that delta can be back propagated to the pooled position

6.11.1.2 int timlCNNPoolingParams::scaleCol

pooling kernel col size

6.11.1.3 int timlCNNPoolingParams::scaleRow

pooling kernel row size

6.11.1.4 int timlCNNPoolingParams::strideX

pooling kernel stride (horizontal)

6.11.1.5 int timlCNNPoolingParams::strideY

pooling kernel stride (vertical)

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

- timlCNN.h

6.12 timlCNNTrainingParams Struct Reference

Data Fields

- int count
- int batchCount
- int epoch
- timlUtilPhase **phase**
- timlUtilAllocatorLevel **allocatorLevel**
- int batchSize
- float **momentum**
- float **learningRate**
- float **weightDecay**
- timlUtilCostFunctionType costType

6.12.1 Field Documentation

6.12.1.1 int timlCNNTrainingParams::batchCount

batch count

6.12.1.2 int timlCNNTrainingParams::batchSize

how many samples do we process until we update the parameters

6.12.1.3 timlUtilCostFunctionType timlCNNTrainingParams::costType

how to evaluate the cost with output of the cnn

6.12.1.4 int timlCNNTrainingParams::count

data count

6.12.1.5 int timlCNNTrainingParams::epoch

how many iterations we need to run through the whole database

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

- timlCNN.h

6.13 timlUtilImage Struct Reference

Data Fields

- float ∗ **data**
- int **row**
- int **col**
- int **channel**

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

- timlUtil.h

6.14 timlUtilImageSet Struct Reference

Data Fields

- float ∗ **data**
- int ∗ **label**
- int **row**
- int **col**
- int **channel**
- int num
- float ∗ mean

6.14.1 Field Documentation

6.14.1.1 float∗ timlUtilImageSet::mean

mean of all the images

6.14.1.2 int timlUtilImageSet::num

number of images

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

- timlUtil.h

6.15 timlUtilInitializer Struct Reference

Data Fields

- timlUtilInitializerType **type**
- float val
- float min
- float max
- float mean
- float std

6.15.1 Field Documentation

6.15.1.1 float timlUtilInitializer::max

uniform initializer

6.15.1.2 float timlUtilInitializer::mean

Gaussian initializer

6.15.1.3 float timlUtilInitializer::min

uniform initializer

6.15.1.4 float timlUtilInitializer::std

Gaussian initializer

6.15.1.5 float timlUtilInitializer::val

constant initializer

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

- timlUtil.h

Chapter 7

File Documentation

7.1 appCNNClass.h File Reference

```
#include "timl.h"
```

Functions

- int appCNNClassMNISTTraining ()
  MNIST training example.
- int appCNNClassMNISTTesting ()
  MNIST classification testing example.
- int appCNNClassCIFAR10Training ()
  CIFAR10 training example.
- int appCNNClassCIFAR10Testing ()
  CIFAR10 testing example.
- int appCNNClassImageNetCaffeNetTesting ()
  CaffeNet classification testing example.
- int appCNNClassImageNetCaffeNetTraining ()
  CaffeNet training example.
- int appCNNClassImageNetAlexNetTesting ()
  AlexNet classification testing example.
- int appCNNClassImageNetVGGNetTesting ()
  VGGNet classification testing example.

7.2 appCNNClassCIFAR10Testing.c File Reference

```
#include "../appCNNClass.h"
```

Macros

- #define **MODEL_PATH** "../../../../database/model/cifar10/databaseModelCIFAR10.m"
- #define **DATABASE_PATH** "../../../../database/cifar10"
- #define **TOP_N** 1
- #define **IMAGE_ROW** 32
- #define **IMAGE_COL** 32
- #define **IMAGE_CHANNEL** 3

Functions

- int **main** ()
- int appCNNClassCIFAR10Testing ()

  CIFAR10 testing example.

7.3 appCNNClassCIFAR10Training.c File Reference

```
#include "../appCNNClass.h"
```

Macros

- #define **DATABASE_PATH** "../../../../database/cifar10"
- #define **IMAGE_ROW** 32
- #define **IMAGE_COL** 32
- #define **IMAGE_CHANNEL** 3
- #define **BATCH_SIZE** 100

Functions

- int **main** ()
- int appCNNClassCIFAR10Training ()

  CIFAR10 training example.

7.4 appCNNClassImageNetAlexNetTesting.c File Reference

```
#include "../appCNNClass.h"
```

Macros

- #define **MODEL_PATH** "../../../../database/model/alexnet/databaseModelAlexNet.m"
- #define **LABEL_PATH** "../../../../database/imagenet/test/label.txt"
- #define **IMAGE_PATH** "../../../../database/imagenet/test/%010d.jpg"
- #define **TOP_N** 5
- #define **IMAGE_NUM** 100
- #define **IMAGE_ROW** 256
- #define **IMAGE_COL** 256
- #define **IMAGE_CHANNEL** 3

Functions

- int **main** ()
- int appCNNClassImageNetAlexNetTesting ()

  AlexNet classification testing example.

7.5 appCNNClassImageNetCaffeNetTesting.c File Reference

```
#include "../appCNNClass.h"
```

Macros

- #define **MODEL_PATH** "../../../../database/model/caffenet/databaseModelCaffeNet.m"
- #define **LABEL_PATH** "../../../../database/imagenet/test/label.txt"
- #define **IMAGE_PATH** "../../../../database/imagenet/test/%010d.jpg"
- #define **TOP_N** 5
- #define **IMAGE_NUM** 100
- #define **IMAGE_ROW** 256
- #define **IMAGE_COL** 256
- #define **IMAGE_CHANNEL** 3

Functions

- int **main** ()
- int appCNNClassImageNetCaffeNetTesting ()

  CaffeNet classification testing example.

7.6 appCNNClassImageNetCaffeNetTraining.c File Reference

```
#include "../appCNNClass.h"
```

Macros

- #define **LABEL_PATH** "../../../../database/imagenet/train/label.txt"
- #define **IMAGE_PATH** "../../../../database/imagenet/train/%010d.jpg"
- #define **TOP_N** 5
- #define **IMAGE_NUM** 100
- #define **IMAGE_BATCH_SIZE** 10
- #define **IMAGE_ROW** 256
- #define **IMAGE_COL** 256
- #define **IMAGE_CROP_ROW** 227
- #define **IMAGE_CROP_COL** 227
- #define **IMAGE_CHANNEL** 3

Functions

- int **main** ()
- int appCNNClassImageNetCaffeNetTraining ()

  CaffeNet training example.

7.7 appCNNClassImageNetVGGNetTesting.c File Reference

```
#include "../appCNNClass.h"
```

Macros

- #define **MODEL_PATH** "../../../../database/model/vggnet/databaseModelVGGNet.m"
- #define **LABEL_PATH** "../../../../database/imagenet/test/label.txt"
- #define **IMAGE_PATH** "../../../../database/imagenet/test/%010d.jpg"
- #define **TOP_N** 5
- #define **IMAGE_NUM** 100
- #define **IMAGE_ROW** 256
- #define **IMAGE_COL** 256
- #define **IMAGE_CHANNEL** 3

Functions

- int **main** ()
- int appCNNClassImageNetVGGNetTesting ()

  VGGNet classification testing example.

7.8 appCNNClassMNISTTesting.c File Reference

```
#include "../appCNNClass.h"
```

Macros

- #define **DATABASE_PATH** "../../../../database/mnist"
- #define **MODEL_PATH** "../../../../database/model/mnist/databaseModelMNIST.m"
- #define **TOP_N** 1
- #define **TEST_NUM** 10000
- #define **IMAGE_ROW** 28
- #define **IMAGE_COL** 28
- #define **IMAGE_CHANNEL** 1

Functions

- int **main** ()
- int appCNNClassMNISTTesting ()

  MNIST classification testing example.

7.9 appCNNClassMNISTTraining.c File Reference

```
#include "../appCNNClass.h"
```

Macros

- #define **DATABASE_PATH** "../../../../database/mnist"
- #define **TRAIN_NUM** 60000
- #define **IMAGE_ROW** 28
- #define **IMAGE_COL** 28
- #define **BATCH_SIZE** 100
- #define **IMAGE_CHANNEL** 1
- #define **LEARN_RATE** 0.1

Functions

- int **main** ()
- int appCNNClassMNISTTraining ()

  MNIST training example.

7.10 appCNNConvertImageNet.cpp File Reference

```
#include "appCNNConvertImageNet.hpp"
```

Functions

- int main (int argc, char ∗argv[ ])

  Convert ImageNet database to have uniform size 256∗256.

7.11 appCNNConvertImageNet.hpp File Reference

```
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <stdio.h>
#include <sys/stat.h>
#include <unistd.h>
#include <cstdlib>
```

Macros

- #define **IMAGE_ROW** 256
- #define **IMAGE_COL** 256
- #define **RAND_SEED** 1
- #define **NAME_BUFFER_SIZE** 50
- #define **IMAGENET_2012_TRAIN_NUM** 1281167
- #define **IMAGENET_2012_TRAIN_CONVERT_FOLDER_MODE** 0777
- #define **IMAGENET_2012_VAL_NUM** 50000
- #define **IMAGENET_2012_VAL_CONVERT_FOLDER_MODE** 0777

Enumerations

- enum **appCNNConvertError** { **ERROR_APP_CNN_CONVERT_ARG**, **ERROR_APP_CNN_CONVERT_IMAGE_NUM**, **ERROR_APP_CNN_CONVERT_ARG**, **ERROR_APP_CNN_CONVERT_IMAGE_NUM** }

Functions

- int appCNNConvertImageNetShuffle (char ∗∗names, int ∗labels, int n)

  Shuffle the images.

7.12 appCNNConvertSBD.cpp File Reference

```
#include "appCNNConvertSBD.hpp"
```

Functions

- int main (int argc, char ∗argv[ ])

  Convert Stanford Background database to have uniform size 240∗320.

7.13 appCNNConvertSBD.hpp File Reference

```
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <stdio.h>
#include <sys/stat.h>
#include <unistd.h>
#include <cstdlib>
```

Macros

- #define **IMAGE_ROW** 240
- #define **IMAGE_COL** 320
- #define **RAND_SEED** 1
- #define **NAME_BUFFER_SIZE** 50
- #define **SBD_IMAGE_NUM** 715
- #define **SBD_TRAIN_NUM** 572
- #define **SBD_TRAIN_CONVERT_FOLDER_MODE** 0777
- #define **SBD_TEST_NUM** 143
- #define **SBD_TEST_CONVERT_FOLDER_MODE** 0777

Enumerations

- enum **appCNNConvertError** { **ERROR_APP_CNN_CONVERT_ARG**, **ERROR_APP_CNN_CONVERT_IMAGE_NUM**, **ERROR_APP_CNN_CONVERT_ARG**, **ERROR_APP_CNN_CONVERT_IMAGE_NUM** }

Functions

- int appCNNConvertSBDShuffle (char ∗∗names, int n)

  Shuffle the images.

7.14 appCNNConvertSBDShuffle.cpp File Reference

```
#include "appCNNConvertSBD.hpp"
```

Functions

- int appCNNConvertSBDShuffle (char ∗∗names, int n)

  Shuffle the images.

7.15 appCNNInteropCaffe.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Macros

- #define **FLOAT_FORMAT** "%12.6f"
- #define **INT_FORMAT** "%5d"

Functions

- int main (int argc, char ∗argv[ ])

  Caffe to TIML CNN model converter.

7.16 appCNNInteropCaffe.hpp File Reference

```
#include <google/protobuf/io/coded_stream.h>
#include <google/protobuf/io/zero_copy_stream_impl.h>
#include <google/protobuf/message.h>
#include <google/protobuf/text_format.h>
#include <fcntl.h>
#include "appCNNInteropCaffeProtobuf.hpp"
#include "timl.h"
```

Enumerations

- enum **appCNNInteropError** { **ERROR_APP_CNN_INTEROP_ARG**, **ERROR_APP_CNN_INTEROP_READ_FIEL** }

Functions

- bool appCNNInteropCaffeReadProtoFromTextFile (const char ∗fileName, Message ∗proto)

  Caffe read proto from text file.
- bool appCNNInteropCaffeReadProtoFromBinaryFile (const char ∗fileName, Message ∗proto)

  Caffe read proto from binary file.
- int appCNNInteropCaffeFlipMatrixFloat (float ∗a, int m, int n)

  Flip a matrix.
- int appCNNInteropCaffeFlipKernelMatrix (float ∗kernel, int kernelRow, int kernelCol, int inputChannel, int outputChannel)

  Flip the kernels.
- int appCNNInteropCaffeFillBlockDiagonalMatrix (float ∗a, int M, int N, int group, float ∗b)

  Fill a block diagonal matrix.
- timlUtilActivationType appCNNInteropCaffeNonlinearTypeConvert (LayerParameter_LayerType type)

Caffe nonlinear layer type conversion.

- timlCNNLayerType appCNNInteropCaffeLayerTypeConvert (LayerParameter_LayerType type)

  Caffe to TIML CNN layer type conversion.
- timlCNNPoolingType appCNNInteropCaffePoolingTypeConvert (PoolingParameter_PoolMethod method)

  Caffe pooling type conversion.
- timlConvNeuralNetwork ∗ appCNNInteropCaffeConvert (const char ∗netStructurePrototxtFileName, const char ∗netParamPrototxtFileName)

  Convert Caffe to TIML CNN.
- int appCNNInteropCaffeConvLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Convert Caffe conv layer.
- int appCNNInteropCaffeConvLayerPermuteKernel (timlCNNLayer ∗layer)

  Change the kernel from BGR squence to RGB.
- int appCNNInteropCaffePoolingLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe pooling layer conversion.
- int appCNNInteropCaffeNormLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe norm layer conversion.
- int appCNNInteropCaffeLinearLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe linear layer conversion.
- int appCNNInteropCaffeNonlinearLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe nonlinear layer conversion.
- int appCNNInteropCaffeDropoutLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe dropout layer conversion.
- int appCNNInteropCaffeReadMean (timlCNNLayer ∗layer, const char ∗fileName)

  Read Caffe mean binary file.
- int appCNNInteropCaffePermuteMean (float ∗mean, int row, int col, int channel)

  Permute the mean in the input layer from BGR sequence to RGB.

7.17 appCNNInteropCaffeConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- timlConvNeuralNetwork ∗ appCNNInteropCaffeConvert (const char ∗netStructurePrototxtFileName, const char ∗netParamPrototxtFileName)

  Convert Caffe to TIML CNN.

7.18 appCNNInteropCaffeConvLayerConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeConvLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Convert Caffe conv layer.

7.19 appCNNInteropCaffeConvLayerPermuteKernel.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeConvLayerPermuteKernel (timlCNNLayer ∗layer)

  Change the kernel from BGR squence to RGB.

7.20 appCNNInteropCaffeDropoutLayerConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeDropoutLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe dropout layer conversion.

7.21 appCNNInteropCaffeFillBlockDiagonalMatrix.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeFillBlockDiagonalMatrix (float ∗a, int M, int N, int group, float ∗b)

  Fill a block diagonal matrix.

7.22 appCNNInteropCaffeFlipKernelMatrix.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeFlipKernelMatrix (float ∗kernel, int kernelRow, int kernelCol, int inputChannel, int outputChannel)

  Flip the kernels.

7.23 appCNNInteropCaffeFlipMatrixFloat.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeFlipMatrixFloat (float ∗a, int m, int n)

  Flip a matrix.

7.24 appCNNInteropCaffeLayerTypeConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- timlCNNLayerType appCNNInteropCaffeLayerTypeConvert (LayerParameter_LayerType type)

  Caffe to TIML CNN layer type conversion.

7.25 appCNNInteropCaffeLinearLayerConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeLinearLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe linear layer conversion.

7.26 appCNNInteropCaffeNonlinearLayerConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeNonlinearLayerConvert (timlConvNeuralNetwork ∗cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe nonlinear layer conversion.

7.27 appCNNInteropCaffeNonlinearTypeConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- timlUtilActivationType appCNNInteropCaffeNonlinearTypeConvert (LayerParameter_LayerType type)

  Caffe nonlinear layer type conversion.

7.28 appCNNInteropCaffeNormLayerConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeNormLayerConvert (timlConvNeuralNetwork *cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe norm layer conversion.

7.29 appCNNInteropCaffePermuteMean.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffePermuteMean (float *mean, int row, int col, int channel)

  Permute the mean in the input layer from BGR sequence to RGB.

7.30 appCNNInteropCaffePoolingLayerConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffePoolingLayerConvert (timlConvNeuralNetwork *cnn, LayerParameter layerStructure, LayerParameter layerParam)

  Caffe pooling layer conversion.

7.31 appCNNInteropCaffePoolingTypeConvert.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- timlCNNPoolingType appCNNInteropCaffePoolingTypeConvert (PoolingParameter_PoolMethod method)

  Caffe pooling type conversion.

7.32 appCNNInteropCaffeReadMean.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- int appCNNInteropCaffeReadMean (timlCNNLayer ∗layer, const char ∗fileName)

  Read Caffe mean binary file.

7.33 appCNNInteropCaffeReadProtoFromBinaryFile.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Macros

- #define **APP_CNN_INTEROP_CAFFE_READ_BINARY_TOTAL_BYTE_LIMIT** 1073741824
- #define **APP_CNN_INTEROP_CAFFE_READ_BINARY_WARNING_THRESHOLD** 1073741824

Functions

- bool appCNNInteropCaffeReadProtoFromBinaryFile (const char ∗fileName, Message ∗proto)

  Caffe read proto from binary file.

7.34 appCNNInteropCaffeReadProtoFromTextFile.cpp File Reference

```
#include "appCNNInteropCaffe.hpp"
```

Functions

- bool appCNNInteropCaffeReadProtoFromTextFile (const char ∗fileName, Message ∗proto)

  Caffe read proto from text file.

7.35 appCNNScene.h File Reference

```
#include "timl.h"
```

Data Structures

- struct appCNNSceneDataSet

Functions

- float appCNNSceneAccuracy (int ∗labelMatrix, int ∗trueLabelMatrix, int dim)

  Return the labeling accuracy.
- int appCNNSceneSupervisedTraining (timlConvNeuralNetwork ∗cnn, appCNNSceneDataSet ∗dataSet)

  Supervised training on the dataset.
- int appCNNSceneSBDTraining ()

  Scene labeling training example.
- int appCNNSceneSBDTesting ()

  Standford Backgournd Database Scene labeling testing example.
- int appCNNSceneClassify (timlConvNeuralNetwork ∗cnn, timlUtilImage image, int ∗labelMatrix, int scale)

  Pixel label classification.
- int appCNNSceneShuffleIdx (int ∗imageIdx, int ∗rowIdx, int ∗colIdx, appCNNSceneDataSet ∗dataSet)

  Shuffles the (image, row, col) index combination from the data set.
- int appCNNSceneGetLabel (int imageIdx, int rowIdx, int colIdx, appCNNSceneDataSet ∗dataSet)

  Return the pixel label for (image, row, col) index combination.
- int appCNNSceneGetPatch (int imageIdx, int rowIdx, int colIdx, appCNNSceneDataSet ∗dataSet, float ∗patch)

  Return the image patch for (image, row, col) index combination.
- int appCNNSceneClassifyOpenMP (timlConvNeuralNetwork ∗∗cnnTeam, int teamNum, float ∗data, int row, int col, int channel, int ∗labelMatrix, int scale)

  Supervised training on the dataset using openmp.
- int appCNNSceneLabelMatrix (float ∗map, int row, int col, int channel, int m, int k, int ∗labelMatrix, int numRow, int numCol)

  Fill the label matrix.

7.36 appCNNSceneAccuracy.c File Reference

```
#include "appCNNScene.h"
```

Functions

- float appCNNSceneAccuracy (int ∗labelMatrix, int ∗trueLabelMatrix, int dim)

  Return the labeling accuracy.

7.37 appCNNSceneClassify.c File Reference

```
#include "appCNNScene.h"
```

Functions

- int appCNNSceneClassify (timlConvNeuralNetwork ∗cnn, timlUtilImage image, int ∗labelMatrix, int scale)

  Pixel label classification.

7.38 appCNNSceneClassifyOpenMP.c File Reference

```
#include "appCNNScene.h"
```

Functions

- int appCNNSceneClassifyOpenMP (timlConvNeuralNetwork ∗∗cnnTeam, int teamNum, float ∗data, int row, int col, int channel, int ∗labelMatrix, int scale)

    Supervised training on the dataset using openmp.

7.39 appCNNSceneGetLabel.c File Reference

```
#include "appCNNScene.h"
```

Functions

- int appCNNSceneGetLabel (int imageIdx, int rowIdx, int colIdx, appCNNSceneDataSet ∗dataSet)

    Return the pixel label for (image, row, col) index combination.

7.40 appCNNSceneGetPatch.c File Reference

```
#include "appCNNScene.h"
```

Functions

- int appCNNSceneGetPatch (int imageIdx, int rowIdx, int colIdx, appCNNSceneDataSet ∗dataSet, float ∗patch)

    Return the image patch for (image, row, col) index combination.

7.41 appCNNSceneLabelMatrix.c File Reference

```
#include "appCNNScene.h"
```

Functions

- int appCNNSceneLabelMatrix (float ∗map, int row, int col, int channel, int m, int k, int ∗labelMatrix, int numRow, int numCol)

    Fill the label matrix.

7.42 appCNNSceneSBDTesting.c File Reference

```
#include "../appCNNScene.h"
```

Macros

- #define **SCALE** 4
- #define **IMAGE_NUM** 10

- #define **IMAGE_ROW** 240
- #define **IMAGE_COL** 320
- #define **IMAGE_CHANNEL** 3
- #define **PATCH_SIZE** 133
- #define **MODEL_PATH** "../../../../database/model/sbd/databaseModelSBD.m"
- #define **IMAGE_PATH** "../../../../database/sbd/test/%03d.jpg"
- #define **LABEL_PATH** "../../../../database/sbd/test/%03d.txt"

Functions

- int **main** ()
- int appCNNSceneSBDTesting ()

  Standford Backgournd Database Scene labeling testing example.

7.43 appCNNSceneSBDTraining.c File Reference

```
#include "../appCNNScene.h"
```

Macros

- #define **IMAGE_ROW** 240
- #define **IMAGE_COL** 320
- #define **IMAGE_CHANNEL** 3
- #define **PATCH_SIZE** 133
- #define **TRAIN_IMAGE_PATH** "../../../../database/sbd/train/%03d.jpg"
- #define **TRAIN_IMAGE_NUM** 450
- #define **TRAIN_LABEL_PATH** "../../../../database/sbd/train/%03d.txt"

Functions

- int **main** ()
- int appCNNSceneSBDTraining ()

  Scene labeling training example.

7.44 appCNNSceneShuffleIdx.c File Reference

```
#include "appCNNScene.h"
```

Functions

- int appCNNSceneShuffleIdx (int ∗imageIdx, int ∗rowIdx, int ∗colIdx, appCNNSceneDataSet ∗dataSet)

  Shuffles the (image, row, col) index combination from the data set.

7.45 appCNNSceneSupervisedTraining.c File Reference

```
#include "appCNNScene.h"
```

Functions

- int appCNNSceneSupervisedTraining (timlConvNeuralNetwork ∗cnn, appCNNSceneDataSet ∗dataSet)

    Supervised training on the dataset.

7.46 benchmarkCNNClass.h File Reference

```
#include "timl.h"
```

Functions

- int benchmarkCNNClassCaffeNetTesting ()

    CNN CaffeNet classification benchmark.
- int benchmarkCNNClassVGGNetTesting ()

    CNN VGGNet classification benchmark.

7.47 benchmarkCNNClassCaffeNetTesting.c File Reference

```
#include "../benchmarkCNNClass.h"
```

Macros

- #define **MODEL_PATH** "../../../../database/model/caffenet/databaseModelCaffeNet.m"
- #define **LABEL_PATH** "../../../../database/imagenet/test/label.txt"
- #define **IMAGE_PATH** "../../../../database/imagenet/test/%010d.jpg"
- #define **TOP_N** 5
- #define **IMAGE_NUM** 100
- #define **IMAGE_ROW** 256
- #define **IMAGE_COL** 256
- #define **IMAGE_CHANNEL** 3
- #define **ITER** 10

Functions

- int **main** ()
- int benchmarkCNNClassCaffeNetTesting ()

    CNN CaffeNet classification benchmark.

7.48 benchmarkCNNClassVGGNetTesting.c File Reference

```
#include "../benchmarkCNNClass.h"
```

Macros

- #define **MODEL_PATH** "../../../../database/model/vggnet/databaseModelVGGNet.m"
- #define **LABEL_PATH** "../../../../database/imagenet/test/label.txt"
- #define **IMAGE_PATH** "../../../../database/imagenet/test/%010d.jpg"
- #define **TOP_N** 5
- #define **IMAGE_NUM** 100
- #define **IMAGE_ROW** 256
- #define **IMAGE_COL** 256
- #define **IMAGE_CHANNEL** 3
- #define **ITER** 10

Functions

- int **main** ()
- int benchmarkCNNClassVGGNetTesting ()

  CNN VGGNet classification benchmark.

7.49 testCNN.c File Reference

```
#include "testCNN.h"
```

Functions

- int **main** ()

7.50 testCNN.h File Reference

```
#include "timl.h"
```

Functions

- int testCNNSimpleTraining ()

  Simple training function test.
- int testCNNSimpleClone ()

  Simple clone function test.
- int testCNNSimpleIO ()

  Simple read/write function test.
- int testCNNSimpleResize ()

  Simple resize function test.
- int testCNNSimpleProfile ()

  simple profile function test

7.51 testCNNSimpleClone.c File Reference

```
#include "testCNN.h"
```

Macros

- #define **MODEL_PATH** "../../database/test/cnn/databaseTestCNNSimpleClone1.m"
- #define **CLONE_MODEL_PATH** "../../database/test/cnn/databaseTestCNNSimpleClone2.m"
- #define **IMAGE_ROW** 28
- #define **IMAGE_COL** 28
- #define **IMAGE_CHANNEL** 3
- #define **INT_FORMAT** "%10d"
- #define **FLOAT_FORMAT** "%12.4f"
- #define **CNN_NAME** "cnn"
- #define **PARAMS_LEVEL** Util_ParamsLevel2

Functions

- int testCNNSimpleClone ()

  Simple clone function test.

7.52 testCNNSimpleIO.c File Reference

```
#include "testCNN.h"
```

Macros

- #define **MODEL_PATH** "../../database/test/cnn/databaseTestCNNSimpleWrite1.m"
- #define **WRITEBACK_MODEL_PATH** "../../database/test/cnn/databaseTestCNNSimpleWrite2.m"
- #define **IMAGE_ROW** 28
- #define **IMAGE_COL** 28
- #define **IMAGE_CHANNEL** 3
- #define **INT_FORMAT** "%10d"
- #define **FLOAT_FORMAT** "%12.4f"
- #define **CNN_NAME** "cnn"
- #define **PARAMS_LEVEL** Util_ParamsLevel3

Functions

- int testCNNSimpleIO ()

  Simple read/write function test.

7.53 testCNNSimpleProfile.c File Reference

```
#include "testCNN.h"
```

Macros

- #define **IMAGE_ROW** 28
- #define **IMAGE_COL** 28
- #define **IMAGE_CHANNEL** 1
- #define **ITER** 2
- #define **BATCH_SIZE** 100
- #define **DATABASE_PATH** "../../database/mnist"

Functions

- int testCNNSimpleProfile ()

  simple profile function test

7.54 testCNNSimpleResize.c File Reference

```
#include "testCNN.h"
```

Macros

- #define **MODEL_PATH** "../../database/test/cnn/databaseTestCNNSimpleResize1.m"
- #define **RESIZED_MODEL_PATH** "../../database/test/cnn/databaseTestCNNSimpleResize2.m"
- #define **IMAGE_ROW** 28
- #define **IMAGE_COL** 28
- #define **IMAGE_CHANNEL** 1
- #define **IMAGE_RESIZE_ROW** 56
- #define **IMAGE_RESIZE_COL** 56
- #define **IMAGE_RESIZE_CHANNEL** 1
- #define **INT_FORMAT** "%10d"
- #define **FLOAT_FORMAT** "%12.4f"
- #define **CNN_NAME** "cnn"
- #define **PARAMS_LEVEL** Util_ParamsLevel2

Functions

- int testCNNSimpleResize ()

  Simple resize function test.

7.55 testCNNSimpleShare.c File Reference

```
#include "testCNN.h"
```

Macros

- #define **MODEL_PATH** "../../database/test/cnn/databaseTestCNNSimpleShare1.m"
- #define **SHARED_MODEL_PATH** "../../database/test/cnn/databaseTestCNNSimpleShare2.m"
- #define **IMAGE_ROW** 28
- #define **IMAGE_COL** 28
- #define **IMAGE_CHANNEL** 1
- #define **INT_FORMAT** "%10d"
- #define **FLOAT_FORMAT** "%12.4f"
- #define **CNN_NAME** "cnn"
- #define **PARAMS_LEVEL** Util_ParamsLevel2

Functions

- int testCNNSimpleShare ()

  Simple share function test.

7.56 testCNNSimpleTraining.c File Reference

```
#include "testCNN.h"
```

Macros

- #define **DATABASE_PATH** "../../database/mnist"
- #define **BATCH_SIZE** 100
- #define **TEST_NUM** 10000
- #define **TRAIN_NUM** 60000
- #define **IMAGE_ROW** 28
- #define **IMAGE_COL** 28
- #define **IMAGE_CHANNEL** 1

Functions

- int testCNNSimpleTraining ()

  Simple training function test.

7.57 testUtil.c File Reference

```
#include "testUtil.h"
```

Functions

- int **main** ()

7.58 testUtil.h File Reference

```
#include "timl.h"
```

Functions

- int testUtilBLAS (void)

  BLAS function test.
- int testUtilConv2 (void)

  2d convoultion function test
- int testUtilSort ()

  Sort function test.

7.59 testUtilBLAS.c File Reference

```
#include "testUtil.h"
```

Functions

- int testUtilBLAS (void)

  BLAS function test.

7.60 testUtilConv2.c File Reference

```
#include "testUtil.h"
```

Functions

- int testUtilConv2 (void)

  2d convoultion function test

7.61 testUtilSort.c File Reference

```
#include "testUtil.h"
```

Functions

- int testUtilSort ()

  Sort function test.

7.62 timl.h File Reference

timl public APIs

```
#include "timlUtil.h"
#include "timlCNN.h"
```

Functions

- timlCNNInputParams timlCNNInputParamsDefault ()

  Return the default parameters for the input layer.
- timlCNNConvParams timlCNNConvParamsDefault ()

  Return the default parameters for the convolutional layer.
- timlCNNLinearParams timlCNNLinearParamsDefault ()

  Return the default parameters for the linear layer.
- timlCNNPoolingParams timlCNNPoolingParamsDefault ()

  Return the default parameters for the pooling layer.
- timlCNNNonlinearParams timlCNNNonlinearParamsDefault ()

  Return the default parameters for the nonlinear layer.
- timlCNNNormParams timlCNNNormParamsDefault ()

  Return the default parameters for the norm layer.
- timlCNNTrainingParams timlCNNTrainingParamsDefault ()

  Return the default training parameters.

- timlConvNeuralNetwork ∗ timlCNNCreateConvNeuralNetwork (timlCNNTrainingParams params, int deviceId)

    Create a cnn structure.
- int timlCNNAddInputLayer (timlConvNeuralNetwork ∗cnn, int featureMapRow, int featureMapCol, int featureMapChannel, timlCNNInputParams params)

    Add input layer.
- int timlCNNAddPoolingLayer (timlConvNeuralNetwork ∗cnn, int scaleRow, int scaleCol, int strideX, int strideY, timlCNNPoolingType type, timlCNNPoolingParams params)

    Add pooling layer.
- int timlCNNAddNormLayer (timlConvNeuralNetwork ∗cnn, timlCNNNormParams params)

    Add normalization layer.
- int timlCNNAddConvLayer (timlConvNeuralNetwork ∗cnn, int kernelRow, int kernelCol, int strideX, int strideY, int featureMapChannel, timlCNNConvParams params)

    Add conv layer.
- int timlCNNAddNonlinearLayer (timlConvNeuralNetwork ∗cnn, timlUtilActivationType type)

    Add nonlinear layer.
- int timlCNNAddLinearLayer (timlConvNeuralNetwork ∗cnn, int dim, timlCNNLinearParams params)

    Add linear layer.
- int timlCNNAddDropoutLayer (timlConvNeuralNetwork ∗cnn, float prob)

    Add dropout layer.
- int timlCNNInitialize (timlConvNeuralNetwork ∗cnn)

    Allocate the memory required by the cnn.
- int timlCNNReset (timlConvNeuralNetwork ∗cnn)

    Reset the parameters of the CNN.
- int timlCNNDelete (timlConvNeuralNetwork ∗cnn)

    Free a cnn structure.
- int timlCNNSupervisedTrainingWithLabelBatchMode (timlConvNeuralNetwork ∗cnn, float ∗data, int ∗label, int dim, int num)

    Supervised training with label.
- int timlCNNClassifyTopNBatchMode (timlConvNeuralNetwork ∗cnn, float ∗data, int dim, int num, int ∗label, float ∗percent, int topN)

    Batch classification.
- int timlCNNClassifyTop1SingleMode (timlConvNeuralNetwork ∗cnn, float ∗data, int dim)

    Classify the data.
- int timlCNNSetMode (timlConvNeuralNetwork ∗cnn, timlUtilPhase phase)

    Set the phase (train/test) of the cnn.
- timlConvNeuralNetwork ∗ timlCNNClone (timlConvNeuralNetwork ∗cnn, int deviceId)

    Clone a cnn.
- timlConvNeuralNetwork ∗ timlCNNShareParams (timlConvNeuralNetwork ∗cnn, int deviceId)

    Create a new CNN that shares the parameters with the input CNN.
- long timlCNNMemory (timlConvNeuralNetwork ∗cnn)

    Return the memory in bytes required by the cnn.
- long timlCNNGetParamsNum (timlConvNeuralNetwork ∗cnn)

    Get the number of parameters of the cnn.
- int timlCNNWriteToFile (const char ∗fileName, timlConvNeuralNetwork ∗cnn, timlUtilParamsLevel paramsLevel, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

    Write the cnn to file(s)
- timlConvNeuralNetwork ∗ timlCNNReadFromFile (const char ∗fileName, int deviceId)

    Read CNN from file(s)
- int timlCNNPrint (timlConvNeuralNetwork ∗cnn)

    Print out the information of the cnn.
- int timlCNNProfile (timlConvNeuralNetwork ∗cnn, float ∗data, int dim, int num, int ∗label, int iter)

Profile the CNN with both timing and memory allocation.

- int timlCNNResize (timlConvNeuralNetwork ∗cnn, int row, int col, int channel)

  Resize the feature map sizes to accommodate new input feature map dimensions.
- int timlCNNGetLayerNum (timlConvNeuralNetwork ∗cnn)

  Return the number of layers of the cnn.

7.62.1 Detailed Description

timl public APIs

7.63 timlCNN.h File Reference

```
#include "timlUtil.h"
```

Data Structures

- struct timlCNNPoolingParams
- struct timlCNNLinearParams
- struct timlCNNDataSet
- struct timlCNNConvParams
- struct timlCNNNonlinearParams
- struct timlCNNNormParams
- struct timlCNNInputParams
- struct timlCNNDropoutParams
- struct _timlCNNLayer_
- struct timlCNNTrainingParams
- struct _timlConvNeuralNetwork_

Macros

- #define **ERROR_CNN_OFFSET** 4000

Typedefs

- typedef struct _timlCNNLayer_ **timlCNNLayer**
- typedef struct _timlConvNeuralNetwork_ **timlConvNeuralNetwork**

Enumerations

- enum **timlCNNError** {
  ERROR_CNN_FEATURE_MAP_SIZE = ERROR_CNN_OFFSET, **ERROR_CNN_FEATURE_MAP_CHANNEL**, **ERROR_CNN_ALLOCATION**, **ERROR_CNN_LAYER_ALLOCATION**,
  ERROR_CNN_TEAM_ALLOCATION, **ERROR_CNN_CONV_LAYER_KERNEL_SIZE**, **ERROR_CNN_CONV_LAYER_PAD_SIZE**, **ERROR_CNN_CONV_LAYER_STRIDE_SIZE**,
  ERROR_CNN_POOLING_LAYER_SCALE_SIZE, **ERROR_CNN_POOLING_LAYER_PAD_SIZE**, **ERROR_CNN_POOLING_LAYER_STRIDE_SIZE**, **ERROR_CNN_INPUT_LAYER_PARAMS**,
  ERROR_CNN_LINEAR_LAYER_DIM, **ERROR_CNN_NORM_LAYER_PARAMS**, **ERROR_CNN_DROPOUT_LAYER_PARAMS**, **ERROR_CNN_NULL_PTR**,
  ERROR_CNN_EMPTY, **ERROR_CNN_READ_FILE**, **ERROR_CNN_CLASS** }

- enum **timlCNNNormType** { **CNN_InterChannel**, **CNN_IntraChannel** }
- enum **timlCNNLayerType** {
  CNN_Input, **CNN_Conv**, **CNN_Pooling**, **CNN_Nonlinear**,
  CNN_Linear, **CNN_Norm**, **CNN_Dropout** }
- enum **timlCNNPoolingType** { **CNN_MaxPooling**, **CNN_MeanPooling** }

Functions

- int timlCNNInputShareParams (timlConvNeuralNetwork ∗cnn, timlCNNLayer ∗layer)

  Share the mean with other input layer.
- int timlCNNConvShareParams (timlConvNeuralNetwork ∗cnnShare, timlCNNLayer ∗layer)

  Share the parameters with other conv layer.
- int timlCNNLinearShareParams (timlConvNeuralNetwork ∗cnnShare, timlCNNLayer ∗layer)

  Share the parameters with other linear layer.
- int timlCNNConvInitialize (timlCNNLayer ∗layer)

  Initialize the conv layer.
- int timlCNNLinearInitialize (timlCNNLayer ∗layer)

  Initialize the linear layer.
- int timlCNNNonlinearInitialize (timlCNNLayer ∗layer)

  Initialize the nonlinear layer.
- int timlCNNInputInitialize (timlCNNLayer ∗layer)

  Initialize the input layer.
- int timlCNNPoolingInitialize (timlCNNLayer ∗layer)

  Initialize the pooling layer.
- int timlCNNNormInitialize (timlCNNLayer ∗layer)

  Initialize the norm layer.
- int timlCNNDropoutInitialize (timlCNNLayer ∗layer)

  Initialize the dropout layer.
- int timlCNNBackPropagation (timlConvNeuralNetwork ∗cnn, timlCNNLayer ∗layer)

  Back propagate the gradient from layer to the first layer of the cnn.
- int timlCNNConvBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the conv layer to the previous layer.
- int timlCNNNormBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the norm layer to the previous layer.
- int timlCNNPoolingBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the pooling layer to the previous layer.
- int timlCNNMaxPoolingBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the max pooling layer to the previous layer.
- int timlCNNMeanPoolingBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the mean pooling layer to the previous layer.
- int timlCNNNonlinearBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the nonlinear layer to the previous layer.
- int timlCNNLinearBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the linear layer to the previous layer.
- int timlCNNDropoutBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the dropout layer to the previous layer.
- int timlCNNCostWithLabel (timlConvNeuralNetwork ∗cnn, int label, float ∗cost, timlCNNLayer ∗∗bpStartLayer)

  Calculate the cost based on the cnn output and the label.
- int timlCNNForwardPropagation (timlConvNeuralNetwork ∗cnn, float ∗data, int dim)

  Forward propagate data to the CNN.

- int timlCNNInputForwardPropagation (timlCNNLayer ∗layer, float ∗data, int dim)

    Forward propagate data to the the input layer.
- int timlCNNLinearForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNDropoutForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNNonlinearForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNNormForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNPoolingForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNMaxPoolingForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNMeanPoolingForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNConvForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.
- int timlCNNDeleteConvLayer (timlCNNLayer ∗layer)

    Delete conv layer.
- int timlCNNDeleteInputLayer (timlCNNLayer ∗layer)

    Delete input layer.
- int timlCNNDeleteNonlinearLayer (timlCNNLayer ∗layer)

    Delete nonlinear layer.
- int timlCNNDeleteNormLayer (timlCNNLayer ∗layer)

    Delete norm layer.
- int timlCNNDeletePoolingLayer (timlCNNLayer ∗layer)

    Delete pooling layer.
- int timlCNNDeleteLinearLayer (timlCNNLayer ∗layer)

    Delete linear layer.
- int timlCNNDeleteDropoutLayer (timlCNNLayer ∗layer)

    Delete dropout layer.
- int timlCNNResetConvLayer (timlCNNLayer ∗layer)

    Reset conv layer.
- int timlCNNResetInputLayer (timlCNNLayer ∗layer)

    Reset input layer.
- int timlCNNResetLinearLayer (timlCNNLayer ∗layer)

    Reset linear layer.
- int timlCNNResetNonlinearLayer (timlCNNLayer ∗layer)

    Reset nonlinear layer.
- int timlCNNResetNormLayer (timlCNNLayer ∗layer)

    Reset norm layer.
- int timlCNNResetPoolingLayer (timlCNNLayer ∗layer)

    Reset pooling layer.
- int timlCNNUpdateParams (timlConvNeuralNetwork ∗cnn)

    Update the parameters of the cnn.
- int timlCNNLinearUpdateParams (timlCNNLayer ∗layer)

    Update the parameters of the linear layer.
- int timlCNNConvUpdateParams (timlCNNLayer ∗layer)

    Update the parameters of the conv layer.

- int timlCNNConvWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the conv layer to file(s)
- int timlCNNNonlinearWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the nonlinear layer to file(s)
- int timlCNNNormWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the norm layer to file(s)
- int timlCNNPoolingWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the pooling layer to file(s)
- int timlCNNLinearWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the linear layer to file(s)
- int timlCNNInputWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the input layer to file(s)
- int timlCNNTrainingParamsWriteToFile (FILE ∗fp, timlConvNeuralNetwork ∗cnn, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the training params to file(s)
- int timlCNNDropoutWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the dropout layer to file(s)
- int timlCNNConvReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the conv layer from a Matlab compatible text file.
- int timlCNNTrainingParamsReadFromTextFile (FILE ∗fp, timlConvNeuralNetwork ∗cnn)

  Read the training params from a text file.
- int timlCNNNormReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the nonlinear layer from a text file.
- int timlCNNPoolingReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the pooling layer from a text file.
- int timlCNNNonlinearReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the nonlinear layer from a text file.
- int timlCNNLinearReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the linear layer from a text file.
- int timlCNNDropoutReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the dropout layer from a text file.
- int timlCNNInputReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the input layer from a Matlab compatible text file.
- int timlCNNConvReadFromBinaryFile (FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer)

  Read the conv layer parameters from binary files.
- int timlCNNLinearReadFromBinaryFile (FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer)

  Read the linear layer parameters from binary files.
- int timlCNNInputReadFromBinaryFile (FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer)

  Read the input layer parameters from binary files.
- int timlCNNAssignDevice (timlConvNeuralNetwork ∗cnn, int deviceId, int threadId)

  Assign the cnn to a specific device and thread.
- const char ∗ timlCNNLayerTypeStr (timlCNNLayer ∗layer)

  Return a string that represents the layer type.
- int timlCNNMemPoolSize (timlConvNeuralNetwork ∗cnn)

  Return the memory pool size (byte)

- int timlCNNSupervisedTrainingWithLabelBatchModeOpenMP (timlConvNeuralNetwork ∗cnn, float ∗data, int ∗label, int dim, int num)

  supervised training with label using openmp
- int timlCNNClassifyTopNBatchModeOpenMP (timlConvNeuralNetwork ∗cnn, float ∗data, int dim, int num, int ∗label, float ∗percent, int topN)

  Batch classification using openmp.
- int timlCNNClassifyTopNTeamModeOpenMP (timlConvNeuralNetwork ∗∗cnnTeam, int num, float ∗data, int dim, int ∗label, float ∗percent, int topN)

  Batch classification using openmp.

7.64 timlCNNAddConvLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNAddConvLayer (timlConvNeuralNetwork ∗cnn, int kernelRow, int kernelCol, int strideX, int strideY, int featureMapChannel, timlCNNConvParams params)

  Add conv layer.

7.65 timlCNNAddDropoutLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNAddDropoutLayer (timlConvNeuralNetwork ∗cnn, float prob)

  Add dropout layer.

7.66 timlCNNAddInputLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNAddInputLayer (timlConvNeuralNetwork ∗cnn, int featureMapRow, int featureMapCol, int featureMapChannel, timlCNNInputParams params)

  Add input layer.

7.67 timlCNNAddLinearLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNAddLinearLayer (timlConvNeuralNetwork ∗cnn, int dim, timlCNNLinearParams params)

  Add linear layer.

7.68 timlCNNAddNonlinearLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNAddNonlinearLayer (timlConvNeuralNetwork ∗cnn, timlUtilActivationType type)

  Add nonlinear layer.

7.69 timlCNNAddNormLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNAddNormLayer (timlConvNeuralNetwork ∗cnn, timlCNNNormParams params)

  Add normalization layer.

7.70 timlCNNAddPoolingLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNAddPoolingLayer (timlConvNeuralNetwork ∗cnn, int scaleRow, int scaleCol, int strideX, int strideY, timlCNNPoolingType type, timlCNNPoolingParams params)

  Add pooling layer.

7.71 timlCNNAssignDevice.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNAssignDevice (timlConvNeuralNetwork ∗cnn, int deviceId, int threadId)

  Assign the cnn to a specific device and thread.

7.72 timlCNNBackPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNBackPropagation (timlConvNeuralNetwork ∗cnn, timlCNNLayer ∗layer)

  Back propagate the gradient from layer to the first layer of the cnn.

7.73 timlCNNClassifyTop1SingleMode.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNClassifyTop1SingleMode (timlConvNeuralNetwork ∗cnn, float ∗data, int dim)

  Classify the data.

7.74 timlCNNClassifyTopNBatchMode.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNClassifyTopNBatchMode (timlConvNeuralNetwork ∗cnn, float ∗data, int dim, int num, int ∗label, float ∗percent, int topN)

  Batch classification.

7.75 timlCNNClassifyTopNBatchModeOpenMP.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNClassifyTopNBatchModeOpenMP (timlConvNeuralNetwork ∗cnn, float ∗data, int dim, int num, int ∗label, float ∗percent, int topN)

  Batch classification using openmp.

7.76 timlCNNClassifyTopNTeamModeOpenMP.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNClassifyTopNTeamModeOpenMP (timlConvNeuralNetwork ∗∗cnnTeam, int num, float ∗data, int dim, int ∗label, float ∗percent, int topN)

  Batch classification using openmp.

7.77 timlCNNClone.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlConvNeuralNetwork ∗ timlCNNClone (timlConvNeuralNetwork ∗cnn, int deviceId)

  Clone a cnn.

7.78 timlCNNConvBackPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNConvBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the conv layer to the previous layer.

7.79 timlCNNConvForwardPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNConvForwardPropagation (timlCNNLayer ∗prevLayer)

  Forward propagate form layer to layer->next.

7.80 timlCNNConvInitialize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNConvInitialize (timlCNNLayer ∗layer)

  Initialize the conv layer.

7.81 timlCNNConvParamsDefault.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlCNNConvParams timlCNNConvParamsDefault ()

  Return the default parameters for the convolutional layer.

7.82 timlCNNConvReadFromBinaryFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNConvReadFromBinaryFile (FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer)

  Read the conv layer parameters from binary files.

7.83 timlCNNConvReadFromTextFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNConvReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the conv layer from a Matlab compatible text file.

7.84 timlCNNConvShareParams.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNConvShareParams (timlConvNeuralNetwork ∗cnnShare, timlCNNLayer ∗layer)

  Share the parameters with other conv layer.

7.85 timlCNNConvUpdateParams.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNConvUpdateParams (timlCNNLayer ∗layer)

  Update the parameters of the conv layer.

7.86 timlCNNConvWriteToFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNConvWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the conv layer to file(s)

7.87 timlCNNCostWithLabel.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNCostWithLabel (timlConvNeuralNetwork ∗cnn, int label, float ∗cost, timlCNNLayer ∗∗bpStartLayer)

  Calculate the cost based on the cnn output and the label.

7.88 timlCNNCreateConvNeuralNetwork.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlConvNeuralNetwork ∗ timlCNNCreateConvNeuralNetwork (timlCNNTrainingParams params, int deviceId)

  Create a cnn structure.

7.89 timlCNNDelete.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDelete (timlConvNeuralNetwork ∗cnn)

  Free a cnn structure.

7.90 timlCNNDeleteConvLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDeleteConvLayer (timlCNNLayer ∗layer)

    Delete conv layer.

7.91 timlCNNDeleteDropoutLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDeleteDropoutLayer (timlCNNLayer ∗layer)

    Delete dropout layer.

7.92 timlCNNDeleteInputLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDeleteInputLayer (timlCNNLayer ∗layer)

    Delete input layer.

7.93 timlCNNDeleteLinearLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDeleteLinearLayer (timlCNNLayer ∗layer)

    Delete linear layer.

7.94 timlCNNDeleteNonlinearLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDeleteNonlinearLayer (timlCNNLayer ∗layer)

  Delete nonlinear layer.

7.95 timlCNNDeleteNormLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDeleteNormLayer (timlCNNLayer ∗layer)

  Delete norm layer.

7.96 timlCNNDeletePoolingLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDeletePoolingLayer (timlCNNLayer ∗layer)

  Delete pooling layer.

7.97 timlCNNDropoutBackPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDropoutBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the dropout layer to the previous layer.

7.98 timlCNNDropoutForwardPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDropoutForwardPropagation (timlCNNLayer ∗prevLayer)

  Forward propagate form layer to layer->next.

7.99 timlCNNDropoutInitialize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDropoutInitialize (timlCNNLayer ∗layer)

  Initialize the dropout layer.

7.100 timlCNNDropoutReadFromTextFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDropoutReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the dropout layer from a text file.

7.101 timlCNNDropoutWriteToFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNDropoutWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the dropout layer to file(s)

7.102 timlCNNForwardPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNForwardPropagation (timlConvNeuralNetwork ∗cnn, float ∗data, int dim)

  Forward propagate data to the CNN.

7.103 timlCNNGetLayerNum.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNGetLayerNum (timlConvNeuralNetwork ∗cnn)

    Return the number of layers of the cnn.

7.104 timlCNNGetParamsNum.c File Reference

```
#include "../api/timl.h"
```

Functions

- long timlCNNGetParamsNum (timlConvNeuralNetwork ∗cnn)

    Get the number of parameters of the cnn.

7.105 timlCNNInitialize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNInitialize (timlConvNeuralNetwork ∗cnn)

    Allocate the memory required by the cnn.

7.106 timlCNNInputInitialize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNInputInitialize (timlCNNLayer ∗layer)

    Initialize the input layer.

7.107 timlCNNInputParamsDefault.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlCNNInputParams timlCNNInputParamsDefault ()

    Return the default parameters for the input layer.

7.108 timlCNNInputReadFromBinaryFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNInputReadFromBinaryFile (FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer)

  Read the input layer parameters from binary files.

7.109 timlCNNInputReadFromTextFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNInputReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the input layer from a Matlab compatible text file.

7.110 timlCNNInputShareParams.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNInputShareParams (timlConvNeuralNetwork ∗cnn, timlCNNLayer ∗layer)

  Share the mean with other input layer.

7.111 timlCNNInputWriteToFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNInputWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the input layer to file(s)

7.112 timlCNNLayerTypeStr.c File Reference

```
#include "../api/timl.h"
```

Functions

- const char ∗ timlCNNLayerTypeStr (timlCNNLayer ∗layer)

  Return a string that represents the layer type.

7.113 timlCNNLinearBackPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNLinearBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the linear layer to the previous layer.

7.114 timlCNNLinearForwardPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNLinearForwardPropagation (timlCNNLayer ∗prevLayer)

  Forward propagate form layer to layer->next.

7.115 timlCNNLinearInitialize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNLinearInitialize (timlCNNLayer ∗layer)

  Initialize the linear layer.

7.116 timlCNNLinearParamsDefault.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlCNNLinearParams timlCNNLinearParamsDefault ()

  Return the default parameters for the linear layer.

7.117 timlCNNLinearReadFromBinaryFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNLinearReadFromBinaryFile (FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer)

  Read the linear layer parameters from binary files.

7.118 timlCNNLinearReadFromTextFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNLinearReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the linear layer from a text file.

7.119 timlCNNLinearShareParams.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNLinearShareParams (timlConvNeuralNetwork ∗cnnShare, timlCNNLayer ∗layer)

  Share the parameters with other linear layer.

7.120 timlCNNLinearUpdateParams.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNLinearUpdateParams (timlCNNLayer ∗layer)

  Update the parameters of the linear layer.

7.121 timlCNNLinearWriteToFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNLinearWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

    Write the linear layer to file(s)

7.122 timlCNNMaxPoolingBackPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNMaxPoolingBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the max pooling layer to the previous layer.

7.123 timlCNNMaxPoolingForwardPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNMaxPoolingForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.

7.124 timlCNNMeanPoolingBackPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNMeanPoolingBackPropagation (timlCNNLayer ∗layer)

    Back propagate the gradient from the mean pooling layer to the previous layer.

7.125 timlCNNMeanPoolingForwardPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNMeanPoolingForwardPropagation (timlCNNLayer ∗prevLayer)

    Forward propagate form layer to layer->next.

7.126 timlCNNMemory.c File Reference

```
#include "../api/timl.h"
```

Functions

- long timlCNNMemory (timlConvNeuralNetwork ∗cnn)

  Return the memory in bytes required by the cnn.

7.127 timlCNNMemPoolSize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNMemPoolSize (timlConvNeuralNetwork ∗cnn)

  Return the memory pool size (byte)

7.128 timlCNNNonlinearBackPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNNonlinearBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the nonlinear layer to the previous layer.

7.129 timlCNNNonlinearForwardPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNNonlinearForwardPropagation (timlCNNLayer ∗prevLayer)

  Forward propagate form layer to layer->next.

7.130 timlCNNNonlinearInitialize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNNonlinearInitialize (timlCNNLayer ∗layer)

  Initialize the nonlinear layer.

7.131 timlCNNNonlinearParamsDefault.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlCNNNonlinearParams timlCNNNonlinearParamsDefault ()

  Return the default parameters for the nonlinear layer.

7.132 timlCNNNonlinearReadFromTextFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNNonlinearReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the nonlinear layer from a text file.

7.133 timlCNNNonlinearWriteToFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNNonlinearWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the nonlinear layer to file(s)

7.134 timlCNNNormBackPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNNormBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the norm layer to the previous layer.

7.135 timlCNNNormForwardPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNNormForwardPropagation (timlCNNLayer ∗prevLayer)

  Forward propagate form layer to layer->next.

7.136 timlCNNNormInitialize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNNormInitialize (timlCNNLayer ∗layer)

  Initialize the norm layer.

7.137 timlCNNNormParamsDefault.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlCNNNormParams timlCNNNormParamsDefault ()

  Return the default parameters for the norm layer.

7.138 timlCNNNormWriteToFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNNormWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the norm layer to file(s)

7.139 timlCNNPoolingBackPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNPoolingBackPropagation (timlCNNLayer ∗layer)

  Back propagate the gradient from the pooling layer to the previous layer.

7.140 timlCNNPoolingForwardPropagation.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNPoolingForwardPropagation (timlCNNLayer ∗prevLayer)

  Forward propagate form layer to layer->next.

7.141 timlCNNPoolingInitialize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNPoolingInitialize (timlCNNLayer ∗layer)

  Initialize the pooling layer.

7.142 timlCNNPoolingParamsDefault.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlCNNPoolingParams timlCNNPoolingParamsDefault ()

  Return the default parameters for the pooling layer.

7.143 timlCNNPoolingReadFromTextFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNPoolingReadFromTextFile (FILE ∗fp1, timlConvNeuralNetwork ∗cnn)

  Read the pooling layer from a text file.

7.144 timlCNNPoolingWriteToFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNPoolingWriteToFile (FILE ∗fp1, FILE ∗fp2, FILE ∗fp3, timlCNNLayer ∗layer, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the pooling layer to file(s)

7.145 timlCNNPrint.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNPrint (timlConvNeuralNetwork ∗cnn)

  Print out the information of the cnn.

7.146 timlCNNProfile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNProfile (timlConvNeuralNetwork ∗cnn, float ∗data, int dim, int num, int ∗label, int iter)

  Profile the CNN with both timing and memory allocation.

7.147 timlCNNReadFromFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlConvNeuralNetwork ∗ timlCNNReadFromFile (const char ∗fileName, int deviceId)

  Read CNN from file(s)

7.148 timlCNNReset.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNReset (timlConvNeuralNetwork ∗cnn)

  Reset the parameters of the CNN.

7.149 timlCNNResetConvLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNResetConvLayer (timlCNNLayer ∗layer)

  Reset conv layer.

7.150 timlCNNResetDropoutLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNResetDropoutLayer (timlCNNLayer ∗layer)

  Reset dropout layer.

7.151 timlCNNResetInputLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNResetInputLayer (timlCNNLayer ∗layer)

  Reset input layer.

7.152 timlCNNResetLinearLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNResetLinearLayer (timlCNNLayer ∗layer)

  Reset linear layer.

7.153 timlCNNResetNonlinearLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNResetNonlinearLayer (timlCNNLayer ∗layer)

  Reset nonlinear layer.

7.154 timlCNNResetNormLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNResetNormLayer (timlCNNLayer ∗layer)

  Reset norm layer.

7.155 timlCNNResetPoolingLayer.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNResetPoolingLayer (timlCNNLayer ∗layer)

  Reset pooling layer.

7.156 timlCNNResize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNResize (timlConvNeuralNetwork ∗cnn, int row, int col, int channel)

  Resize the feature map sizes to accommodate new input feature map dimensions.

7.157 timlCNNSetMode.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNSetMode (timlConvNeuralNetwork ∗cnn, timlUtilPhase phase)

  Set the phase (train/test) of the cnn.

7.158 timlCNNShareParams.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlConvNeuralNetwork ∗ timlCNNShareParams (timlConvNeuralNetwork ∗cnn, int deviceId)

  Create a new CNN that shares the parameters with the input CNN.

7.159 timlCNNSupervisedTrainingWithLabelBatchMode.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNSupervisedTrainingWithLabelBatchMode (timlConvNeuralNetwork ∗cnn, float ∗data, int ∗label, int dim, int num)

  Supervised training with label.

7.160 timlCNNSupervisedTrainingWithLabelBatchModeOpenMP.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNSupervisedTrainingWithLabelBatchModeOpenMP (timlConvNeuralNetwork ∗cnn, float ∗data, int ∗label, int dim, int num)

  supervised training with label using openmp

7.161 timlCNNTrainingParamsDefault.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlCNNTrainingParams timlCNNTrainingParamsDefault ()

  Return the default training parameters.

7.162 timlCNNTrainingParamsReadFromTextFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNTrainingParamsReadFromTextFile (FILE ∗fp, timlConvNeuralNetwork ∗cnn)

  Read the training params from a text file.

7.163 timlCNNTrainingParamsWriteToFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNTrainingParamsWriteToFile (FILE ∗fp, timlConvNeuralNetwork ∗cnn, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the training params to file(s)

7.164 timlCNNUpdateParams.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNUpdateParams (timlConvNeuralNetwork ∗cnn)

  Update the parameters of the cnn.

7.165 timlCNNWriteToFile.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlCNNWriteToFile (const char ∗fileName, timlConvNeuralNetwork ∗cnn, timlUtilParamsLevel level, const char ∗name, const char ∗floatFormat, const char ∗intFormat)

  Write the cnn to file(s)

7.166 timlUtil.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include <float.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <dirent.h>
#include <omp.h>
#include <unistd.h>
#include <libgen.h>
#include "jpeglib.h"
#include "cblas.h"
```

Data Structures

- struct timlUtilImage
- struct timlUtilInitializer
- struct timlUtilImageSet

Macros

- #define **TIML_UTIL_MAX_STR** 100
- #define **TIML_UTIL_PI** 3.14159265358979323846
- #define **ERROR_UTIL_OFFSET** 3000

Enumerations

- enum **timlUtilError** {
  ERROR_UTIL_NULL_PTR = ERROR_UTIL_OFFSET, **ERROR_UTIL_MNIST_TRAINING_DATA_READING**, **ERROR_UTIL_MNIST_TRAINING_DATA_ALLOCATION**, **ERROR_UTIL_MNIST_TRAINING_LABEL_READING**,
  ERROR_UTIL_MNIST_TRAINING_LABEL_ALLOCATION, **ERROR_UTIL_MNIST_TESTING_DATA_READING**, **ERROR_UTIL_MNIST_TESTING_DATA_ALLOCATION**, **ERROR_UTIL_MNIST_TESTING_LABEL_READING**,
  ERROR_UTIL_MNIST_TESTING_LABEL_ALLOCATION, **ERROR_UTIL_CIFAR10_TRAINING_READING**, **ERROR_UTIL_CIFAR10_TRAINING_ALLOCATION**, **ERROR_UTIL_CIFAR10_TESTING_READING**,
  ERROR_UTIL_CIFAR10_TESTING_ALLOCATION, **ERROR_UTIL_CIFAR100_TRAINING_READING**, **ERROR_UTIL_CIFAR100_TRAINING_ALLOCATION**, **ERROR_UTIL_CIFAR100_TESTING_READING**,
  ERROR_UTIL_CIFAR100_TESTING_ALLOCATION, **ERROR_UTIL_READ_FLOAT_MATRIX**, **ERROR_UTIL_READ_INT_MATRIX**, **ERROR_UTIL_READ_FLOAT_VECTOR**,
  ERROR_UTIL_READ_INT_VECTOR, **ERROR_UTIL_WRITE_FLOAT_MATRIX**, **ERROR_UTIL_WRITE_INT_MATRIX**, **ERROR_UTIL_WRITE_FLOAT_VECTOR**,
  ERROR_UTIL_WRITE_INT_VECTOR, **ERROR_UTIL_MALLOC**, **ERROR_UTIL_JPEG_READING** }
- enum **timlUtilActivationType** {
  Util_Sigmoid, **Util_Softmax**, **Util_Softplus**, **Util_Relu**,
  Util_Nrelu, **Util_Tanh**, **Util_Linear** }
- enum **timlUtilCostFunctionType** { **Util_CrossEntropy**, **Util_MSE** }
- enum **timlUtilConvType** { **Util_Conv2D**, **Util_Corr2D** }
- enum timlUtilParamsLevel { Util_ParamsLevel1, Util_ParamsLevel2, Util_ParamsLevel3 }
- enum timlUtilAllocatorLevel { Util_AllocatorLevel1, Util_AllocatorLevel2, Util_AllocatorLevel3 }

- enum timlUtilCropType { Util_CenterCrop, Util_RandomCrop }
- enum timlUtilMirrorType { Util_Mirror, Util_NoMirror, Util_RandomMirror }
- enum **timlUtilInitializerType** { **Util_Constant**, **Util_Gaussian**, **Util_Uniform**, **Util_Xavier** }
- enum **timlUtilPhase** { **Util_Train**, **Util_Test**, **Util_Debug** }

Functions

- int timlUtilReadMNIST (const char ∗path, timlUtilImageSet ∗training, timlUtilImageSet ∗testing)

  Read MNIST database from binary files.
- int timlUtilReadCIFAR10 (const char ∗path, timlUtilImageSet ∗training, timlUtilImageSet ∗testing)

  Read CIFA10 database from binary files.
- long timlUtilDiffTime (struct timespec start, struct timespec end)

  Return the time difference in micro second.
- int timlUtilRandDiscreteUniformRNG (int a, int b)

  Discrete uniform random number generator in [a, b].
- int timlUtilRandContinuousUniformRNG (float ∗x, int dim, float a, float b)

  Generate a discrete uniform random vector between (a, b)
- int timlUtilRandNormalRNG (float ∗x, int dim, float mean, float std)

  Generate a Gaussian random number.
- int timlUtilRandPerm (int ∗array, int n)

  Random permute an array.
- int timlUtilFread (void ∗ptr, size_t size, size_t nmemb, FILE ∗fp)

  Read binary file.
- int timlUtilFwrite (const void ∗ptr, size_t size, size_t nmemb, FILE ∗fp)

  Write to a binar file.
- int timlUtilConv2Valid (float ∗a, float ∗b, float ∗c, int aRow, int aCol, int bRow, int bCol)

  conv2(a, b, 'valid')
- int timlUtilConv2Full (float ∗a, float ∗b, float ∗c, int aRow, int aCol, int bRow, int bCol)

  conv2(a, b, 'full')
- int timlUtilCorr2Full (float ∗a, float ∗b, float ∗c, int aRow, int aCol, int bRow, int bCol)

  conv2(a, rot90(b,2), 'valid')
- int timlUtilConv2ImageReshapeBack (float ∗x, float ∗xReshape, int ∗index, int channel, int xDim, int indexDim, int deviceId, int threadId)

  Reshape the convolution matrix back to feature maps.
- int timlUtilConv2ImageReshapeIndex (int ∗index, int aRow, int aCol, int bRow, int bCol, int padUp, int padDown, int padLeft, int padRight, int strideX, int strideY, timlUtilConvType type)

  Create a reshaping index matrix.
- int timlUtilConv2ImageReshape (float ∗xReshape, float ∗x, int ∗index, int channel, int xDim, int indexDim, int deviceId, int threadId)

  Reshape feature maps to a format that turns 2d convolution to GEMM operation.
- timlUtilImage timlUtilReadJPEG (const char ∗name)

  read a jpg image
- int timlUtilReadFixedSizeJPEG (const char ∗name, float ∗data, int row, int col, int channel)

  Read a jpg image with known size information to avoid frequent allocation and deallocation of data.
- char ∗∗ timlUtilScanJPEG (const char ∗dirName, int ∗imageNum)

  Return an array of jpg image names in the directory.
- void timlUtilBLASdgemm (const enum CBLAS_TRANSPOSE TransA, const enum CBLAS_TRANSPOSE TransB, const int M, const int N, const int K, const double alpha, const double ∗A, const double ∗B, const double beta, double ∗C, int deviceId, int threadId)

  Double general matrix matrix multiplication C = alpha ∗ op(A) ∗ op(B) + beta ∗ C op(A) : M∗K op(B) : K∗N.

- void timlUtilBLASsgemm (const enum CBLAS_TRANSPOSE TransA, const enum CBLAS_TRANSPOSE TransB, const int M, const int N, const int K, const float alpha, const float ∗A, const float ∗B, const float beta, float ∗C, int deviceId, int threadId)

  Float general matrix matrix multiplication C = alpha ∗ op(A) ∗ op(B) + beta ∗ C op(A) : M∗K op(B) : K∗N.
- void timlUtilBLASdgemv (const enum CBLAS_TRANSPOSE TransA, const int M, const int N, const double alpha, const double ∗A, const double ∗x, const double beta, double ∗y, int deviceId, int threadId)

  Double general matrix vector multiplication y = alpha ∗ op(A) ∗ x + beta ∗ y op(A): M∗N.
- void timlUtilBLASsgemv (const enum CBLAS_TRANSPOSE TransA, const int M, const int N, const float alpha, const float ∗A, const float ∗x, const float beta, float ∗y, int deviceId, int threadId)

  Float general matrix vector multiplication y = alpha ∗ op(A) ∗ x + beta ∗ y op(A): M∗N.
- void timlUtilBLASsaxpy (const int N, const float alpha, const float ∗X, float ∗Y, int deviceId, int threadId)

  Float vector addition Y = alpha ∗ X + Y.
- void timlUtilBLASdaxpy (const int N, const double alpha, const double ∗X, double ∗Y, int deviceId, int threadId)

  Double vector addition Y = alpha ∗ X + Y.
- void timlUtilBLASscopy (const int N, const float ∗X, float ∗Y, int deviceId, int threadId)

  Float vector copy Y = X.
- void timlUtilBLASdcopy (const int N, const double ∗X, double ∗Y, int deviceId, int threadId)

  Double vector copy Y = X.
- void timlUtilBLASsger (const int M, const int N, const float alpha, float ∗x, float ∗y, float ∗A, int deviceId, int threadId)

  Float vector outer product A = alpha∗x∗y' + A; x: M y: N.
- void timlUtilBLASdger (const int M, const int N, const double alpha, double ∗x, double ∗y, double ∗A, int deviceId, int threadId)

  Double vector outer product A = alpha∗x∗y' + A; x: M y: N.
- void timlUtilBLASdscal (const int N, const double alpha, double ∗X, int deviceId, int threadId)

  Double vector scaling x = alpha ∗ x.
- void timlUtilBLASsscal (const int N, const float alpha, float ∗X, int deviceId, int threadId)

  Float vector scaling x = alpha ∗ x.
- int timlUtilVectorResetFloat (float ∗a, int m, float val, int deviceId, int threadId)

  Reset a float vector.
- int timlUtilVectorResetInt (int ∗a, int m, int val, int deviceId, int threadId)

  Reset an int vector.
- float timlUtilVectorSumFloat (float ∗a, int n)

  Calculate the sum of a float vector.
- int timlUtilVectorSortFloat (float ∗a, int n)

  Sort an array in descending order.
- int timlUtilVectorSortIndexFloat (float ∗a, int ∗index, int n)

  Sort an array in descending order and return the indices of the original elements in the sorted array.
- float timlUtilVectorMaxFloat (float ∗x, int n, int inc)

  Return the max value in the array.
- int timlUtilVectorMaxIndexFloat (float ∗x, int n, int inc)

  Return the max value index in the array.
- int timlUtilElementWiseMultiply (float ∗a, const float ∗b, const float ∗c, int dim, int deviceId, int threadId)

  Element wise multiply c = a.∗b.
- int timlUtilSubtract (float ∗x, float y, int deviceId, int threadId)

  Subtract operation.
- int timlUtilSigmoid (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Sigmoid.
- int timlUtilSigmoidDerivative (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Sigmoid derivative.
- int timlUtilRelu (float ∗x, float ∗y, int n, int deviceId, int threadId)

Rectified linear unit.

- int timlUtilReluDerivative (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Rectified linear unit derivative.
- int timlUtilTanhDerivative (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Tanh derivative.
- float timlUtilMultinomialCrossEntropy (float ∗x, int label, int n)

  Calculate the mutlinomial cross entropy between x and label.
- float timlUtilMeanSqaureError (float ∗x, int label, int n)

  Calculate the mean square error between x and label.
- int timlUtilSoftmax (float ∗x, float ∗y, int row, int col, int channel, int deviceId, int threadId)

  Softmax function.
- int timlUtilClassifyAccuracy (int ∗label, int topN, int num, int ∗trueLabel)

  Calculate the classification accuracy.
- void timlUtilTransform (float ∗dataOut, float ∗dataIn, float ∗dataHost, int channel, int row, int col, int rowOffset, int colOffset, int rowIn, int colIn, float scale, float ∗mean, timlUtilMirrorType mirrorType, int deviceId, int threadId)

  Transform the raw input data with preprocessing.
- int timlUtilMaxPooling (float ∗outputMap, int ∗maxIndex, float ∗inputMap, int row, int col, int channel, int prevRow, int prevCol, int scaleRow, int scaleCol, int padUp, int padLeft, int strideX, int strideY, timlUtilPhase phase, int deviceId, int threadId)

  Max pooling.
- int timlUtilUndoMaxPooling (float ∗prevDelta, int ∗maxIndex, float ∗delta, int dim, int deviceId, int threadId)

  Undo max pooling.
- int timlUtilMeanPooling (float ∗outputMap, float ∗inputMap, int row, int col, int channel, int prevRow, int prevCol, int scaleRow, int scaleCol, int padUp, int padLeft, int strideX, int strideY, int deviceId, int threadId)

  Mean pooling.
- int timlUtilUndoMeanPooling (float ∗prevDelta, float ∗delta, int row, int col, int channel, int prevRow, int prevCol, int scaleRow, int scaleCol, int padUp, int padLeft, int strideX, int strideY, int deviceId, int threadId)

  Undo mean pooling.
- int timlUtilLocalContrastNormalize (float ∗inputMap, float ∗outputMap, float ∗denom, int row, int col, int channel, int N, float alpha, float beta, int deviceId, int threadId)

  Local contrast normalization.
- int timlUtilLocalContrastUnnormalize (float ∗prevDelta, float ∗prevFeatureMap, float ∗delta, float ∗featureMap, float ∗denom, int row, int col, int channel, int N, float alpha, float beta, int deviceId, int threadId)

  Local contrast unnormalization.
- int timlUtilMasking (float ∗inputMap, float ∗outputMap, int ∗mask, unsigned int ∗randomVector, int dim, float prob, int deviceId, int threadId)

  Masking feature maps.
- int timlUtilUnmasking (float ∗inputDelta, float ∗outputDelta, int ∗mask, int dim, float prob, int deviceId, int threadId)

  Masking feature maps.
- int timlUtilTanh (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Tanh.
- int timlUtilMalloc (void ∗∗devPtr, size_t size)

  memory allocation
- void timlUtilFree (void ∗ptr)

  Free pointer.
- uint32_t timlUtilReverseEndian32 (register uint32_t i)

  Reverse the 32 bit endian pattern.
- int timlUtilElementWiseFunction (float ∗x, float ∗y, int n, float(∗func)(float))

  Apply a function on each element of the array.

7.167 timlUtilBLAS.c File Reference

```
#include "../api/timl.h"
```

Functions

- void timlUtilBLASdgemm (const enum CBLAS_TRANSPOSE TransA, const enum CBLAS_TRANSPOSE TransB, const int M, const int N, const int K, const double alpha, const double ∗A, const double ∗B, const double beta, double ∗C, int deviceId, int threadId)

  Double general matrix matrix multiplication C = alpha ∗ op(A) ∗ op(B) + beta ∗ C op(A) : M∗K op(B) : K∗N.
- void timlUtilBLASsgemm (const enum CBLAS_TRANSPOSE TransA, const enum CBLAS_TRANSPOSE TransB, const int M, const int N, const int K, const float alpha, const float ∗A, const float ∗B, const float beta, float ∗C, int deviceId, int threadId)

  Float general matrix matrix multiplication C = alpha ∗ op(A) ∗ op(B) + beta ∗ C op(A) : M∗K op(B) : K∗N.
- void timlUtilBLASdgemv (const enum CBLAS_TRANSPOSE TransA, const int M, const int N, const double alpha, const double ∗A, const double ∗x, const double beta, double ∗y, int deviceId, int threadId)

  Double general matrix vector multiplication y = alpha ∗ op(A) ∗ x + beta ∗ y op(A): M∗N.
- void timlUtilBLASsgemv (const enum CBLAS_TRANSPOSE TransA, const int M, const int N, const float alpha, const float ∗A, const float ∗x, const float beta, float ∗y, int deviceId, int threadId)

  Float general matrix vector multiplication y = alpha ∗ op(A) ∗ x + beta ∗ y op(A): M∗N.
- void timlUtilBLASsaxpy (const int N, const float alpha, const float ∗X, float ∗Y, int deviceId, int threadId)

  Float vector addition Y = alpha ∗ X + Y.
- void timlUtilBLASdaxpy (const int N, const double alpha, const double ∗X, double ∗Y, int deviceId, int threadId)

  Double vector addition Y = alpha ∗ X + Y.
- void timlUtilBLASscopy (const int N, const float ∗X, float ∗Y, int deviceId, int threadId)

  Float vector copy Y = X.
- void timlUtilBLASdcopy (const int N, const double ∗X, double ∗Y, int deviceId, int threadId)

  Double vector copy Y = X.
- void timlUtilBLASdger (const int M, const int N, const double alpha, double ∗x, double ∗y, double ∗A, int deviceId, int threadId)

  Double vector outer product A = alpha∗x∗y' + A; x: M y: N.
- void timlUtilBLASsger (const int M, const int N, const float alpha, float ∗x, float ∗y, float ∗A, int deviceId, int threadId)

  Float vector outer product A = alpha∗x∗y' + A; x: M y: N.
- void timlUtilBLASdscal (const int N, const double alpha, double ∗X, int deviceId, int threadId)

  Double vector scaling x = alpha ∗ x.
- void timlUtilBLASsscal (const int N, const float alpha, float ∗X, int deviceId, int threadId)

  Float vector scaling x = alpha ∗ x.

7.168 timlUtilClassifyAccuracy.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilClassifyAccuracy (int ∗label, int topN, int num, int ∗trueLabel)

  Calculate the classification accuracy.

7.169 timlUtilConv2Full.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilConv2Full (float ∗a, float ∗b, float ∗c, int aRow, int aCol, int bRow, int bCol)

  conv2(a, b, 'full')

7.170 timlUtilConv2ImageReshape.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilConv2ImageReshape (float ∗xReshape, float ∗x, int ∗index, int channel, int xDim, int indexDim, int deviceId, int threadId)

  Reshape feature maps to a format that turns 2d convolution to GEMM operation.

7.171 timlUtilConv2ImageReshapeBack.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilConv2ImageReshapeBack (float ∗x, float ∗xReshape, int ∗index, int channel, int xDim, int indexDim, int deviceId, int threadId)

  Reshape the convolution matrix back to feature maps.

7.172 timlUtilConv2ImageReshapeIndex.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilConv2ImageReshapeIndex (int ∗index, int aRow, int aCol, int bRow, int bCol, int padUp, int padDown, int padLeft, int padRight, int strideX, int strideY, timlUtilConvType type)

  Create a reshaping index matrix.

7.173 timlUtilConv2Valid.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilConv2Valid (float ∗a, float ∗b, float ∗c, int aRow, int aCol, int bRow, int bCol)

  conv2(a, b, 'valid')

7.174 timlUtilCorr2Full.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilCorr2Full (float ∗a, float ∗b, float ∗c, int aRow, int aCol, int bRow, int bCol)

  conv2(a, rot90(b,2), 'valid')

7.175 timlUtilDiffTime.c File Reference

```
#include "../api/timl.h"
```

Functions

- long timlUtilDiffTime (struct timespec start, struct timespec end)

  Return the time difference in micro second.

7.176 timlUtilElementWiseFunction.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilElementWiseFunction (float ∗x, float ∗y, int n, float(∗func)(float))

  Apply a function on each element of the array.

7.177 timlUtilElementWiseMultiply.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilElementWiseMultiply (float ∗a, const float ∗b, const float ∗c, int dim, int deviceId, int threadId)

  Element wise multiply c = a.∗b.

7.178 timlUtilFread.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilFread (void ∗ptr, size_t size, size_t nmemb, FILE ∗fp)

  Read binary file.

7.179 timlUtilFree.c File Reference

```
#include "../api/timl.h"
```

Functions

- void timlUtilFree (void ∗ptr)

  Free pointer.

7.180 timlUtilFwrite.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilFwrite (const void ∗ptr, size_t size, size_t nmemb, FILE ∗fp)

  Write to a binar file.

7.181 timlUtilLocalContrastNormalize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilLocalContrastNormalize (float ∗inputMap, float ∗outputMap, float ∗denom, int row, int col, int channel, int N, float alpha, float beta, int deviceId, int threadId)

  Local contrast normalization.

7.182 timlUtilLocalContrastUnnormalize.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilLocalContrastUnnormalize (float ∗prevDelta, float ∗prevFeatureMap, float ∗delta, float ∗featureMap, float ∗denom, int row, int col, int channel, int N, float alpha, float beta, int deviceId, int threadId)

  Local contrast unnormalization.

7.183 timlUtilMalloc.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilMalloc (void ∗∗devPtr, size_t size)

  memory allocation

7.184 timlUtilMasking.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilMasking (float ∗inputMap, float ∗outputMap, int ∗mask, unsigned int ∗randomVector, int dim, float prob, int deviceId, int threadId)

  Masking feature maps.

7.185 timlUtilMaxPooling.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilMaxPooling (float ∗outputMap, int ∗maxIndex, float ∗inputMap, int row, int col, int channel, int prevRow, int prevCol, int scaleRow, int scaleCol, int padUp, int padLeft, int strideX, int strideY, timlUtilPhase phase, int deviceId, int threadId)

  Max pooling.

7.186 timlUtilMeanPooling.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilMeanPooling (float ∗outputMap, float ∗inputMap, int row, int col, int channel, int prevRow, int prevCol, int scaleRow, int scaleCol, int padUp, int padLeft, int strideX, int strideY, int deviceId, int threadId)

  Mean pooling.

7.187 timlUtilMeanSquareError.c File Reference

```
#include "../api/timl.h"
```

Functions

- float timlUtilMeanSqaureError (float ∗x, int label, int n)

  Calculate the mean square error between x and label.

7.188 timlUtilMultinomialCrossEntropy.c File Reference

```
#include "../api/timl.h"
```

Functions

- float timlUtilMultinomialCrossEntropy (float ∗x, int label, int n)

  Calculate the mutlinomial cross entropy between x and label.

7.189 timlUtilRandContinuousUniformRNG.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilRandContinuousUniformRNG (float ∗x, int dim, float a, float b)

  Generate a discrete uniform random vector between (a, b)

7.190 timlUtilRandDiscreteUniformRNG.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilRandDiscreteUniformRNG (int a, int b)

  Discrete uniform random number generator in [a, b].

7.191 timlUtilRandNormalRNG.c File Reference

```
#include "../api/timl.h"
```

Functions

- float **uniformRNG** ()
- int timlUtilRandNormalRNG (float ∗x, int dim, float mean, float std)

  Generate a Gaussian random number.

7.192 timlUtilRandPerm.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilRandPerm (int ∗array, int n)

  Random permute an array.

7.193 timlUtilReadCIFAR10.c File Reference

```
#include "../api/timl.h"
```

Macros

- #define **TRAIN_FILE_NUM** 5
- #define **SIZE** 32
- #define **CHANNEL** 3
- #define **SAMPLE_PER_FILE** 10000

Functions

- int timlUtilReadCIFAR10 (const char ∗path, timlUtilImageSet ∗training, timlUtilImageSet ∗testing)

  Read CIFA10 database from binary files.

7.194 timlUtilReadFixedSizeJPEG.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilReadFixedSizeJPEG (const char ∗name, float ∗data, int row, int col, int channel)

  Read a jpg image with known size information to avoid frequent allocation and deallocation of data.

7.195 timlUtilReadJPEG.c File Reference

```
#include "../api/timl.h"
```

Functions

- timlUtilImage timlUtilReadJPEG (const char ∗name)

  read a jpg image

7.196 timlUtilReadMNIST.c File Reference

```
#include "../api/timl.h"
```

Macros

- #define **DATA_MAGIC_NUM** 2051
- #define **LABEL_MAGIC_NUM** 2049
- #define **CHANNEL** 1

Functions

- int timlUtilReadMNIST (const char ∗path, timlUtilImageSet ∗training, timlUtilImageSet ∗testing)

  Read MNIST database from binary files.

7.197 timlUtilRelu.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilRelu (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Rectified linear unit.

7.198 timlUtilReluDerivative.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilReluDerivative (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Rectified linear unit derivative.

7.199 timlUtilReverseEndian32.c File Reference

```
#include "../api/timl.h"
```

Functions

- uint32_t timlUtilReverseEndian32 (register uint32_t i)

  Reverse the 32 bit endian pattern.

7.200 timlUtilScanJPEG.c File Reference

```
#include "../api/timl.h"
```

Functions

- char ∗∗ timlUtilScanJPEG (const char ∗dirName, int ∗imageNum)

  Return an array of jpg image names in the directory.

7.201 timlUtilSigmoid.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilSigmoid (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Sigmoid.

7.202 timlUtilSigmoidDerivative.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilSigmoidDerivative (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Sigmoid derivative.

7.203 timlUtilSoftmax.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilSoftmax (float ∗x, float ∗y, int row, int col, int channel, int deviceId, int threadId)

  Softmax function.

7.204 timlUtilSubtract.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilSubtract (float ∗x, float y, int deviceId, int threadId)

  Subtract operation.

7.205 timlUtilTanh.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilTanh (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Tanh.

7.206 timlUtilTanhDerivative.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilTanhDerivative (float ∗x, float ∗y, int n, int deviceId, int threadId)

  Tanh derivative.

7.207 timlUtilTransform.c File Reference

```
#include "../api/timl.h"
```

Functions

- void timlUtilTransform (float ∗dataOut, float ∗dataIn, float ∗dataHost, int channel, int row, int col, int rowOffset, int colOffset, int rowIn, int colIn, float scale, float ∗mean, timlUtilMirrorType mirrorType, int deviceId, int threadId)

  Transform the raw input data with preprocessing.

7.208 timlUtilUndoMaxPooling.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilUndoMaxPooling (float ∗prevDelta, int ∗maxIndex, float ∗delta, int dim, int deviceId, int threadId)

  Undo max pooling.

7.209 timlUtilUndoMeanPooling.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilUndoMeanPooling (float ∗prevDelta, float ∗delta, int row, int col, int channel, int prevRow, int prevCol, int scaleRow, int scaleCol, int padUp, int padLeft, int strideX, int strideY, int deviceId, int threadId)

  Undo mean pooling.

7.210 timlUtilUnmasking.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilUnmasking (float ∗inputDelta, float ∗outputDelta, int ∗mask, int dim, float prob, int deviceId, int threadId)

  Masking feature maps.

7.211 timlUtilVectorMaxFloat.c File Reference

```
#include "../api/timl.h"
```

Functions

- float timlUtilVectorMaxFloat (float ∗x, int n, int inc)

  Return the max value in the array.

7.212 timlUtilVectorMaxIndexFloat.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilVectorMaxIndexFloat (float ∗x, int n, int inc)

  Return the max value index in the array.

7.213 timlUtilVectorResetFloat.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilVectorResetFloat (float ∗a, int m, float val, int deviceId, int threadId)

  Reset a float vector.

7.214 timlUtilVectorResetInt.c File Reference

```
#include "../api/timl.h"
```

Functions

- int timlUtilVectorResetInt (int ∗a, int m, int val, int deviceId, int threadId)

  Reset an int vector.

7.215 timlUtilVectorSortFloat.c File Reference

```
#include "../api/timl.h"
```

Functions

- int **timlUtilSortCompareFloat** (const void ∗a, const void ∗b)
- int timlUtilVectorSortFloat (float ∗a, int n)

  Sort an array in descending order.

7.216 timlUtilVectorSortIndexFloat.c File Reference

```
#include "../api/timl.h"
```

Functions

- int **timlUtilSortCompareFloatPointer** (const void ∗a, const void ∗b)
- int timlUtilVectorSortIndexFloat (float ∗a, int ∗index, int n)

  Sort an array in descending order and return the indices of the original elements in the sorted array.

7.217 timlUtilVectorSumFloat.c File Reference

```
#include "../api/timl.h"
```

Functions

- float timlUtilVectorSumFloat (float ∗a, int n)

  Calculate the sum of a float vector.

Bibliography

[1] I. Sutskever A. Krizhevsky and G. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems (NIPS)*, pages 1–9, 2012. 4

[2] J. Yangqing et. al. Caffe: convolutional architecture for fast feature embedding. *arXiv:1408.5093*, pages 1–4, 2014. 4

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, page 1–13, 2014. 5

Index

timlCNNLayer, 109
  delta, 109
  dropoutParams, 109
  prev, 109
timlConvNeuralNetwork, 110
  memPool, 110
  memPoolSize, 110

app, 103
appCNN, 104
appCNNClass, 25
appCNNClass.h, 119
appCNNClassCIFAR10Testing.c, 119
appCNNClassCIFAR10Training.c, 120
appCNNClassImageNetAlexNetTesting.c, 120
appCNNClassImageNetCaffeNetTesting.c, 121
appCNNClassImageNetCaffeNetTraining.c, 121
appCNNClassImageNetVGGNetTesting.c, 121
appCNNClassMNISTTesting.c, 122
appCNNClassMNISTTraining.c, 122
appCNNConvertImageNet, 26
  appCNNConvertImageNetShuffle, 26
  main, 26
appCNNConvertImageNet.cpp, 123
appCNNConvertImageNet.hpp, 123
appCNNConvertImageNetShuffle
  appCNNConvertImageNet, 26
appCNNConvertSBD, 27
  appCNNConvertSBDShuffle, 27
  main, 27
appCNNConvertSBD.cpp, 124
appCNNConvertSBD.hpp, 124
appCNNConvertSBDShuffle
  appCNNConvertSBD, 27
appCNNConvertSBDShuffle.cpp, 124
appCNNInteropCaffe, 28
  appCNNInteropCaffeConvLayerConvert, 29
  appCNNInteropCaffeConvLayerPermuteKernel, 29
  appCNNInteropCaffeConvert, 29
  appCNNInteropCaffeDropoutLayerConvert, 29
  appCNNInteropCaffeFillBlockDiagonalMatrix, 30
  appCNNInteropCaffeFlipKernelMatrix, 30
  appCNNInteropCaffeFlipMatrixFloat, 30
  appCNNInteropCaffeLayerTypeConvert, 30
  appCNNInteropCaffeLinearLayerConvert, 32
  appCNNInteropCaffeNonlinearLayerConvert, 32
  appCNNInteropCaffeNonlinearTypeConvert, 32
  appCNNInteropCaffeNormLayerConvert, 32
  appCNNInteropCaffePermuteMean, 33
  appCNNInteropCaffePoolingLayerConvert, 33
  appCNNInteropCaffePoolingTypeConvert, 33
  appCNNInteropCaffeReadMean, 33
  appCNNInteropCaffeReadProtoFromBinaryFile, 34
  appCNNInteropCaffeReadProtoFromTextFile, 34
  main, 34
appCNNInteropCaffe.cpp, 125
appCNNInteropCaffe.hpp, 125
appCNNInteropCaffeConvLayerConvert
  appCNNInteropCaffe, 29
appCNNInteropCaffeConvLayerConvert.cpp, 126
appCNNInteropCaffeConvLayerPermuteKernel
  appCNNInteropCaffe, 29
appCNNInteropCaffeConvLayerPermuteKernel.cpp, 127
appCNNInteropCaffeConvert
  appCNNInteropCaffe, 29
appCNNInteropCaffeConvert.cpp, 126
appCNNInteropCaffeDropoutLayerConvert
  appCNNInteropCaffe, 29
appCNNInteropCaffeDropoutLayerConvert.cpp, 127
appCNNInteropCaffeFillBlockDiagonalMatrix
  appCNNInteropCaffe, 30
appCNNInteropCaffeFillBlockDiagonalMatrix.cpp, 127
appCNNInteropCaffeFlipKernelMatrix
  appCNNInteropCaffe, 30
appCNNInteropCaffeFlipKernelMatrix.cpp, 127
appCNNInteropCaffeFlipMatrixFloat
  appCNNInteropCaffe, 30
appCNNInteropCaffeFlipMatrixFloat.cpp, 128
appCNNInteropCaffeLayerTypeConvert
  appCNNInteropCaffe, 30
appCNNInteropCaffeLayerTypeConvert.cpp, 128
appCNNInteropCaffeLinearLayerConvert
  appCNNInteropCaffe, 32
appCNNInteropCaffeLinearLayerConvert.cpp, 128
appCNNInteropCaffeNonlinearLayerConvert
  appCNNInteropCaffe, 32
appCNNInteropCaffeNonlinearLayerConvert.cpp, 128
appCNNInteropCaffeNonlinearTypeConvert
  appCNNInteropCaffe, 32
appCNNInteropCaffeNonlinearTypeConvert.cpp, 128
appCNNInteropCaffeNormLayerConvert
  appCNNInteropCaffe, 32
appCNNInteropCaffeNormLayerConvert.cpp, 129
appCNNInteropCaffePermuteMean
  appCNNInteropCaffe, 33
appCNNInteropCaffePermuteMean.cpp, 129
appCNNInteropCaffePoolingLayerConvert
  appCNNInteropCaffe, 33

appCNNInteropCaffePoolingLayerConvert.cpp, 129
appCNNInteropCaffePoolingTypeConvert
    appCNNInteropCaffe, 33
appCNNInteropCaffePoolingTypeConvert.cpp, 129
appCNNInteropCaffeReadMean
    appCNNInteropCaffe, 33
appCNNInteropCaffeReadMean.cpp, 130
appCNNInteropCaffeReadProtoFromBinaryFile
    appCNNInteropCaffe, 34
appCNNInteropCaffeReadProtoFromBinaryFile.cpp, 130
appCNNInteropCaffeReadProtoFromTextFile
    appCNNInteropCaffe, 34
appCNNInteropCaffeReadProtoFromTextFile.cpp, 130
appCNNScene, 35
    appCNNSceneAccuracy, 35
    appCNNSceneClassify, 36
    appCNNSceneClassifyOpenMP, 36
    appCNNSceneGetLabel, 36
    appCNNSceneGetPatch, 36
    appCNNSceneLabelMatrix, 38
    appCNNSceneShuffleIdx, 38
    appCNNSceneSupervisedTraining, 38
appCNNScene.h, 130
appCNNSceneAccuracy
    appCNNScene, 35
appCNNSceneAccuracy.c, 131
appCNNSceneClassify
    appCNNScene, 36
appCNNSceneClassify.c, 131
appCNNSceneClassifyOpenMP
    appCNNScene, 36
appCNNSceneClassifyOpenMP.c, 131
appCNNSceneDataSet, 110
    patchSize, 110
appCNNSceneGetLabel
    appCNNScene, 36
appCNNSceneGetLabel.c, 132
appCNNSceneGetPatch
    appCNNScene, 36
appCNNSceneGetPatch.c, 132
appCNNSceneLabelMatrix
    appCNNScene, 38
appCNNSceneLabelMatrix.c, 132
appCNNSceneSBDTesting.c, 132
appCNNSceneSBDTraining.c, 133
appCNNSceneShuffleIdx
    appCNNScene, 38
appCNNSceneShuffleIdx.c, 133
appCNNSceneSupervisedTraining
    appCNNScene, 38
appCNNSceneSupervisedTraining.c, 133

batchCount
    timlCNNTrainingParams, 116
batchSize
    timlCNNTrainingParams, 116
benchmark, 105
benchmarkCNN, 106
benchmarkCNNClass, 40
benchmarkCNNClass.h, 134
benchmarkCNNClassCaffeNetTesting.c, 134
benchmarkCNNClassVGGNetTesting.c, 134

channel
    timlCNNInputParams, 113
channelPermute
    timlCNNInputParams, 113
cnn, 41
    timlCNNAddConvLayer, 46
    timlCNNAddDropoutLayer, 46
    timlCNNAddInputLayer, 47
    timlCNNAddLinearLayer, 47
    timlCNNAddNonlinearLayer, 47
    timlCNNAddNormLayer, 47
    timlCNNAddPoolingLayer, 48
    timlCNNAssignDevice, 48
    timlCNNBackPropagation, 48
    timlCNNClassifyTop1SingleMode, 48
    timlCNNClassifyTopNBatchMode, 49
    timlCNNClassifyTopNBatchModeOpenMP, 49
    timlCNNClassifyTopNTeamModeOpenMP, 49
    timlCNNClone, 50
    timlCNNConvBackPropagation, 50
    timlCNNConvForwardPropagation, 50
    timlCNNConvInitialize, 50
    timlCNNConvParamsDefault, 51
    timlCNNConvReadFromBinaryFile, 51
    timlCNNConvReadFromTextFile, 51
    timlCNNConvShareParams, 51
    timlCNNConvUpdateParams, 52
    timlCNNConvWriteToFile, 52
    timlCNNCostWithLabel, 52
    timlCNNCreateConvNeuralNetwork, 52
    timlCNNDelete, 54
    timlCNNDeleteConvLayer, 54
    timlCNNDeleteDropoutLayer, 54
    timlCNNDeleteInputLayer, 54
    timlCNNDeleteLinearLayer, 54
    timlCNNDeleteNonlinearLayer, 56
    timlCNNDeleteNormLayer, 56
    timlCNNDeletePoolingLayer, 56
    timlCNNDropoutBackPropagation, 56
    timlCNNDropoutForwardPropagation, 56
    timlCNNDropoutInitialize, 58
    timlCNNDropoutReadFromTextFile, 58
    timlCNNDropoutWriteToFile, 58
    timlCNNForwardPropagation, 58
    timlCNNGetLayerNum, 59
    timlCNNGetParamsNum, 59
    timlCNNInitialize, 59
    timlCNNInputForwardPropagation, 59
    timlCNNInputInitialize, 59
    timlCNNInputParamsDefault, 61
    timlCNNInputReadFromBinaryFile, 61
    timlCNNInputReadFromTextFile, 61
    timlCNNInputShareParams, 61
    timlCNNInputWriteToFile, 62

timlCNNLayerTypeStr, 62
timlCNNLinearBackPropagation, 62
timlCNNLinearForwardPropagation, 62
timlCNNLinearInitialize, 63
timlCNNLinearParamsDefault, 63
timlCNNLinearReadFromBinaryFile, 63
timlCNNLinearReadFromTextFile, 63
timlCNNLinearShareParams, 63
timlCNNLinearUpdateParams, 65
timlCNNLinearWriteToFile, 65
timlCNNMaxPoolingBackPropagation, 65
timlCNNMaxPoolingForwardPropagation, 65
timlCNNMeanPoolingBackPropagation, 66
timlCNNMeanPoolingForwardPropagation, 66
timlCNNMemPoolSize, 66
timlCNNMemory, 66
timlCNNNonlinearBackPropagation, 66
timlCNNNonlinearForwardPropagation, 67
timlCNNNonlinearInitialize, 67
timlCNNNonlinearParamsDefault, 67
timlCNNNonlinearReadFromTextFile, 67
timlCNNNonlinearWriteToFile, 67
timlCNNNormBackPropagation, 68
timlCNNNormForwardPropagation, 68
timlCNNNormInitialize, 68
timlCNNNormParamsDefault, 68
timlCNNNormReadFromTextFile, 68
timlCNNNormWriteToFile, 69
timlCNNPoolingBackPropagation, 69
timlCNNPoolingForwardPropagation, 69
timlCNNPoolingInitialize, 69
timlCNNPoolingParamsDefault, 71
timlCNNPoolingReadFromTextFile, 71
timlCNNPoolingWriteToFile, 71
timlCNNPrint, 71
timlCNNProfile, 72
timlCNNReadFromFile, 72
timlCNNReset, 72
timlCNNResetConvLayer, 72
timlCNNResetDropoutLayer, 73
timlCNNResetInputLayer, 73
timlCNNResetLinearLayer, 73
timlCNNResetNonlinearLayer, 73
timlCNNResetNormLayer, 73
timlCNNResetPoolingLayer, 75
timlCNNResize, 75
timlCNNSetMode, 75
timlCNNShareParams, 75
timlCNNSupervisedTrainingWithLabelBatchMode, 76
timlCNNSupervisedTrainingWithLabelBatchModeOpenMP, 76
timlCNNTrainingParamsDefault, 76
timlCNNTrainingParamsReadFromTextFile, 76
timlCNNTrainingParamsWriteToFile, 77
timlCNNUpdateParams, 77
timlCNNWriteToFile, 77
col
timlCNNInputParams, 113
connectivity
timlCNNConvParams, 111
costType
timlCNNTrainingParams, 116
count
timlCNNTrainingParams, 116

data
timlCNNDataSet, 112
delta
timlCNNLayer, 109
denom
timlCNNNormParams, 115
dim
timlCNNLinearParams, 114
dropoutParams
timlCNNLayer, 109

epoch
timlCNNTrainingParams, 116

inputData
timlCNNInputParams, 113

kernel
timlCNNConvParams, 111

label
timlCNNDataSet, 112

main
appCNNConvertImageNet, 26
appCNNConvertSBD, 27
appCNNInteropCaffe, 34
mask
timlCNNDropoutParams, 112
max
timlUtilInitializer, 118
maxIndex
timlCNNPoolingParams, 115
mean
timlCNNInputParams, 113
timlUtilImageSet, 117
timlUtilInitializer, 118
memPool
timlConvNeuralNetwork, 110
memPoolSize
timlConvNeuralNetwork, 110
min
timlUtilInitializer, 118

num
timlUtilImageSet, 117

patchSize
appCNNSceneDataSet, 110
prev
timlCNNLayer, 109
prevDim

timlCNNLinearParams, 114
prevFeatureMapReshape
timlCNNConvParams, 111
prevFeatureMapReshapeIndex
timlCNNConvParams, 111
prob
timlCNNDropoutParams, 112

randomVector
timlCNNDropoutParams, 112
row
timlCNNInputParams, 113

scaleCol
timlCNNPoolingParams, 115
scaleRow
timlCNNPoolingParams, 115
shared
timlCNNInputParams, 113
timlCNNLinearParams, 114
std
timlUtilInitializer, 118
strideX
timlCNNPoolingParams, 115
strideY
timlCNNPoolingParams, 116

test, 107
testCNNSimpleProfile, 107
testCNN, 100
testCNNSimpleClone, 100
testCNNSimpleIO, 100
testCNNSimpleProfile, 100
testCNNSimpleResize, 100
testCNNSimpleShare, 101
testCNNSimpleTraining, 101
testCNN.c, 135
testCNN.h, 135
testCNNSimpleClone
testCNN, 100
testCNNSimpleClone.c, 135
testCNNSimpleIO
testCNN, 100
testCNNSimpleIO.c, 136
testCNNSimpleProfile
test, 107
testCNN, 100
testCNNSimpleProfile.c, 136
testCNNSimpleResize
testCNN, 100
testCNNSimpleResize.c, 137
testCNNSimpleShare
testCNN, 101
testCNNSimpleShare.c, 137
testCNNSimpleTraining
testCNN, 101
testCNNSimpleTraining.c, 138
testUtil, 102
testUtilBLAS, 102
testUtilConv2, 102
testUtilSort, 102
testUtil.c, 138
testUtil.h, 138
testUtilBLAS
testUtil, 102
testUtilBLAS.c, 138
testUtilConv2
testUtil, 102
testUtilConv2.c, 139
testUtilSort
testUtil, 102
testUtilSort.c, 139
timl.h, 139
timlCNN.h, 141
timlCNNAddConvLayer
cnn, 46
timlCNNAddConvLayer.c, 145
timlCNNAddDropoutLayer
cnn, 46
timlCNNAddDropoutLayer.c, 145
timlCNNAddInputLayer
cnn, 47
timlCNNAddInputLayer.c, 145
timlCNNAddLinearLayer
cnn, 47
timlCNNAddLinearLayer.c, 145
timlCNNAddNonlinearLayer
cnn, 47
timlCNNAddNonlinearLayer.c, 146
timlCNNAddNormLayer
cnn, 47
timlCNNAddNormLayer.c, 146
timlCNNAddPoolingLayer
cnn, 48
timlCNNAddPoolingLayer.c, 146
timlCNNAssignDevice
cnn, 48
timlCNNAssignDevice.c, 146
timlCNNBackPropagation
cnn, 48
timlCNNBackPropagation.c, 147
timlCNNClassifyTop1SingleMode
cnn, 48
timlCNNClassifyTop1SingleMode.c, 147
timlCNNClassifyTopNBatchMode
cnn, 49
timlCNNClassifyTopNBatchMode.c, 147
timlCNNClassifyTopNBatchModeOpenMP
cnn, 49
timlCNNClassifyTopNBatchModeOpenMP.c, 147
timlCNNClassifyTopNTeamModeOpenMP
cnn, 49
timlCNNClassifyTopNTeamModeOpenMP.c, 147
timlCNNClone
cnn, 50
timlCNNClone.c, 148
timlCNNConvBackPropagation

cnn, 50
timlCNNConvBackPropagation.c, 148
timlCNNConvForwardPropagation
cnn, 50
timlCNNConvForwardPropagation.c, 148
timlCNNConvInitialize
cnn, 50
timlCNNConvInitialize.c, 148
timlCNNConvParams, 111
connectivity, 111
kernel, 111
prevFeatureMapReshape, 111
prevFeatureMapReshapeIndex, 111
timlCNNConvParamsDefault
cnn, 51
timlCNNConvParamsDefault.c, 149
timlCNNConvReadFromBinaryFile
cnn, 51
timlCNNConvReadFromBinaryFile.c, 149
timlCNNConvReadFromTextFile
cnn, 51
timlCNNConvReadFromTextFile.c, 149
timlCNNConvShareParams
cnn, 51
timlCNNConvShareParams.c, 149
timlCNNConvUpdateParams
cnn, 52
timlCNNConvUpdateParams.c, 149
timlCNNConvWriteToFile
cnn, 52
timlCNNConvWriteToFile.c, 150
timlCNNCostWithLabel
cnn, 52
timlCNNCostWithLabel.c, 150
timlCNNCreateConvNeuralNetwork
cnn, 52
timlCNNCreateConvNeuralNetwork.c, 150
timlCNNDataSet, 112
data, 112
label, 112
timlCNNDelete
cnn, 54
timlCNNDelete.c, 150
timlCNNDeleteConvLayer
cnn, 54
timlCNNDeleteConvLayer.c, 151
timlCNNDeleteDropoutLayer
cnn, 54
timlCNNDeleteDropoutLayer.c, 151
timlCNNDeleteInputLayer
cnn, 54
timlCNNDeleteInputLayer.c, 151
timlCNNDeleteLinearLayer
cnn, 54
timlCNNDeleteLinearLayer.c, 151
timlCNNDeleteNonlinearLayer
cnn, 56
timlCNNDeleteNonlinearLayer.c, 151
timlCNNDeleteNormLayer
cnn, 56
timlCNNDeleteNormLayer.c, 152
timlCNNDeletePoolingLayer
cnn, 56
timlCNNDeletePoolingLayer.c, 152
timlCNNDropoutBackPropagation
cnn, 56
timlCNNDropoutBackPropagation.c, 152
timlCNNDropoutForwardPropagation
cnn, 56
timlCNNDropoutForwardPropagation.c, 152
timlCNNDropoutInitialize
cnn, 58
timlCNNDropoutInitialize.c, 153
timlCNNDropoutParams, 112
mask, 112
prob, 112
randomVector, 112
timlCNNDropoutReadFromTextFile
cnn, 58
timlCNNDropoutReadFromTextFile.c, 153
timlCNNDropoutWriteToFile
cnn, 58
timlCNNDropoutWriteToFile.c, 153
timlCNNForwardPropagation
cnn, 58
timlCNNForwardPropagation.c, 153
timlCNNGetLayerNum
cnn, 59
timlCNNGetLayerNum.c, 153
timlCNNGetParamsNum
cnn, 59
timlCNNGetParamsNum.c, 154
timlCNNInitialize
cnn, 59
timlCNNInitialize.c, 154
timlCNNInputForwardPropagation
cnn, 59
timlCNNInputInitialize
cnn, 59
timlCNNInputInitialize.c, 154
timlCNNInputParams, 113
channel, 113
channelPermute, 113
col, 113
inputData, 113
mean, 113
row, 113
shared, 113
timlCNNInputParamsDefault
cnn, 61
timlCNNInputParamsDefault.c, 154
timlCNNInputReadFromBinaryFile
cnn, 61
timlCNNInputReadFromBinaryFile.c, 155
timlCNNInputReadFromTextFile
cnn, 61

timlCNNInputReadFromTextFile.c, 155
timlCNNInputShareParams
cnn, 61
timlCNNInputShareParams.c, 155
timlCNNInputWriteToFile
cnn, 62
timlCNNInputWriteToFile.c, 155
timlCNNLayerTypeStr
cnn, 62
timlCNNLayerTypeStr.c, 155
timlCNNLinearBackPropagation
cnn, 62
timlCNNLinearBackPropagation.c, 156
timlCNNLinearForwardPropagation
cnn, 62
timlCNNLinearForwardPropagation.c, 156
timlCNNLinearInitialize
cnn, 63
timlCNNLinearInitialize.c, 156
timlCNNLinearParams, 114
dim, 114
prevDim, 114
shared, 114
timlCNNLinearParamsDefault
cnn, 63
timlCNNLinearParamsDefault.c, 156
timlCNNLinearReadFromBinaryFile
cnn, 63
timlCNNLinearReadFromBinaryFile.c, 157
timlCNNLinearReadFromTextFile
cnn, 63
timlCNNLinearReadFromTextFile.c, 157
timlCNNLinearShareParams
cnn, 63
timlCNNLinearShareParams.c, 157
timlCNNLinearUpdateParams
cnn, 65
timlCNNLinearUpdateParams.c, 157
timlCNNLinearWriteToFile
cnn, 65
timlCNNLinearWriteToFile.c, 157
timlCNNMaxPoolingBackPropagation
cnn, 65
timlCNNMaxPoolingBackPropagation.c, 158
timlCNNMaxPoolingForwardPropagation
cnn, 65
timlCNNMaxPoolingForwardPropagation.c, 158
timlCNNMeanPoolingBackPropagation
cnn, 66
timlCNNMeanPoolingBackPropagation.c, 158
timlCNNMeanPoolingForwardPropagation
cnn, 66
timlCNNMeanPoolingForwardPropagation.c, 158
timlCNNMemPoolSize
cnn, 66
timlCNNMemPoolSize.c, 159
timlCNNMemory
cnn, 66
timlCNNMemory.c, 159
timlCNNNonlinearBackPropagation
cnn, 66
timlCNNNonlinearBackPropagation.c, 159
timlCNNNonlinearForwardPropagation
cnn, 67
timlCNNNonlinearForwardPropagation.c, 159
timlCNNNonlinearInitialize
cnn, 67
timlCNNNonlinearInitialize.c, 159
timlCNNNonlinearParams, 114
timlCNNNonlinearParamsDefault
cnn, 67
timlCNNNonlinearParamsDefault.c, 160
timlCNNNonlinearReadFromTextFile
cnn, 67
timlCNNNonlinearReadFromTextFile.c, 160
timlCNNNonlinearWriteToFile
cnn, 67
timlCNNNonlinearWriteToFile.c, 160
timlCNNNormBackPropagation
cnn, 68
timlCNNNormBackPropagation.c, 160
timlCNNNormForwardPropagation
cnn, 68
timlCNNNormForwardPropagation.c, 161
timlCNNNormInitialize
cnn, 68
timlCNNNormInitialize.c, 161
timlCNNNormParams, 115
denom, 115
timlCNNNormParamsDefault
cnn, 68
timlCNNNormParamsDefault.c, 161
timlCNNNormReadFromTextFile
cnn, 68
timlCNNNormWriteToFile
cnn, 69
timlCNNNormWriteToFile.c, 161
timlCNNPoolingBackPropagation
cnn, 69
timlCNNPoolingBackPropagation.c, 161
timlCNNPoolingForwardPropagation
cnn, 69
timlCNNPoolingForwardPropagation.c, 162
timlCNNPoolingInitialize
cnn, 69
timlCNNPoolingInitialize.c, 162
timlCNNPoolingParams, 115
maxIndex, 115
scaleCol, 115
scaleRow, 115
strideX, 115
strideY, 116
timlCNNPoolingParamsDefault
cnn, 71
timlCNNPoolingParamsDefault.c, 162
timlCNNPoolingReadFromTextFile

cnn, 71
timlCNNPoolingReadFromTextFile.c, 162
timlCNNPoolingWriteToFile
cnn, 71
timlCNNPoolingWriteToFile.c, 163
timlCNNPrint
cnn, 71
timlCNNPrint.c, 163
timlCNNProfile
cnn, 72
timlCNNProfile.c, 163
timlCNNReadFromFile
cnn, 72
timlCNNReadFromFile.c, 163
timlCNNReset
cnn, 72
timlCNNReset.c, 163
timlCNNResetConvLayer
cnn, 72
timlCNNResetConvLayer.c, 164
timlCNNResetDropoutLayer
cnn, 73
timlCNNResetDropoutLayer.c, 164
timlCNNResetInputLayer
cnn, 73
timlCNNResetInputLayer.c, 164
timlCNNResetLinearLayer
cnn, 73
timlCNNResetLinearLayer.c, 164
timlCNNResetNonlinearLayer
cnn, 73
timlCNNResetNonlinearLayer.c, 165
timlCNNResetNormLayer
cnn, 73
timlCNNResetNormLayer.c, 165
timlCNNResetPoolingLayer
cnn, 75
timlCNNResetPoolingLayer.c, 165
timlCNNResize
cnn, 75
timlCNNResize.c, 165
timlCNNSetMode
cnn, 75
timlCNNSetMode.c, 165
timlCNNShareParams
cnn, 75
timlCNNShareParams.c, 166
timlCNNSupervisedTrainingWithLabelBatchMode
cnn, 76
timlCNNSupervisedTrainingWithLabelBatchMode.c, 166
timlCNNSupervisedTrainingWithLabelBatchModeOpenMP
cnn, 76
timlCNNSupervisedTrainingWithLabelBatchModeOpenMP.c, 166
timlCNNTrainingParams, 116
batchCount, 116
batchSize, 116
costType, 116
count, 116
epoch, 116
timlCNNTrainingParamsDefault
cnn, 76
timlCNNTrainingParamsDefault.c, 166
timlCNNTrainingParamsReadFromTextFile
cnn, 76
timlCNNTrainingParamsReadFromTextFile.c, 167
timlCNNTrainingParamsWriteToFile
cnn, 77
timlCNNTrainingParamsWriteToFile.c, 167
timlCNNUpdateParams
cnn, 77
timlCNNUpdateParams.c, 167
timlCNNWriteToFile
cnn, 77
timlCNNWriteToFile.c, 167
timlUtil.h, 168
timlUtilAllocatorLevel
util, 81
timlUtilBLAS.c, 172
timlUtilClassifyAccuracy
util, 82
timlUtilClassifyAccuracy.c, 172
timlUtilConv2Full
util, 82
timlUtilConv2Full.c, 173
timlUtilConv2ImageReshape
util, 83
timlUtilConv2ImageReshape.c, 173
timlUtilConv2ImageReshapeBack
util, 83
timlUtilConv2ImageReshapeBack.c, 173
timlUtilConv2ImageReshapeIndex
util, 83
timlUtilConv2ImageReshapeIndex.c, 173
timlUtilConv2Valid
util, 84
timlUtilConv2Valid.c, 173
timlUtilCorr2Full
util, 84
timlUtilCorr2Full.c, 174
timlUtilCropType
util, 81
timlUtilDiffTime
util, 84
timlUtilDiffTime.c, 174
timlUtilElementWiseFunction
util, 86
timlUtilElementWiseFunction.c, 174
timlUtilElementWiseMultiply
util, 86
timlUtilElementWiseMultiply.c, 174
timlUtilFread
util, 86
timlUtilFread.c, 175
timlUtilFree

util, 86
timlUtilFree.c, 175
timlUtilFwrite
util, 87
timlUtilFwrite.c, 175
timlUtilImage, 117
timlUtilImageSet, 117
mean, 117
num, 117
timlUtilInitializer, 117
max, 118
mean, 118
min, 118
std, 118
val, 118
timlUtilLocalContrastNormalize
util, 87
timlUtilLocalContrastNormalize.c, 175
timlUtilLocalContrastUnnormalize
util, 87
timlUtilLocalContrastUnnormalize.c, 175
timlUtilMalloc
util, 88
timlUtilMalloc.c, 176
timlUtilMasking
util, 88
timlUtilMasking.c, 176
timlUtilMaxPooling
util, 88
timlUtilMaxPooling.c, 176
timlUtilMeanPooling
util, 89
timlUtilMeanPooling.c, 176
timlUtilMeanSqaureError
util, 89
timlUtilMeanSquareError.c, 177
timlUtilMirrorType
util, 82
timlUtilMultinomialCrossEntropy
util, 90
timlUtilMultinomialCrossEntropy.c, 177
timlUtilParamsLevel
util, 82
timlUtilRandContinuousUniformRNG
util, 90
timlUtilRandContinuousUniformRNG.c, 177
timlUtilRandDiscreteUniformRNG
util, 90
timlUtilRandDiscreteUniformRNG.c, 177
timlUtilRandNormalRNG
util, 90
timlUtilRandNormalRNG.c, 178
timlUtilRandPerm
util, 91
timlUtilRandPerm.c, 178
timlUtilReadCIFAR10
util, 91
timlUtilReadCIFAR10.c, 178
timlUtilReadFixedSizeJPEG
util, 91
timlUtilReadFixedSizeJPEG.c, 178
timlUtilReadJPEG
util, 91
timlUtilReadJPEG.c, 179
timlUtilReadMNIST
util, 92
timlUtilReadMNIST.c, 179
timlUtilRelu
util, 92
timlUtilRelu.c, 179
timlUtilReluDerivative
util, 92
timlUtilReluDerivative.c, 179
timlUtilReverseEndian32
util, 92
timlUtilReverseEndian32.c, 180
timlUtilScanJPEG
util, 93
timlUtilScanJPEG.c, 180
timlUtilSigmoid
util, 93
timlUtilSigmoid.c, 180
timlUtilSigmoidDerivative
util, 93
timlUtilSigmoidDerivative.c, 180
timlUtilSoftmax
util, 93
timlUtilSoftmax.c, 180
timlUtilSubtract
util, 94
timlUtilSubtract.c, 181
timlUtilTanh
util, 94
timlUtilTanh.c, 181
timlUtilTanhDerivative
util, 94
timlUtilTanhDerivative.c, 181
timlUtilTransform
util, 95
timlUtilTransform.c, 181
timlUtilUndoMaxPooling
util, 95
timlUtilUndoMaxPooling.c, 182
timlUtilUndoMeanPooling
util, 95
timlUtilUndoMeanPooling.c, 182
timlUtilUnmasking
util, 96
timlUtilUnmasking.c, 182
timlUtilVectorMaxFloat
util, 96
timlUtilVectorMaxFloat.c, 182
timlUtilVectorMaxIndexFloat
util, 96
timlUtilVectorMaxIndexFloat.c, 182
timlUtilVectorResetFloat

util, 98
timlUtilVectorResetFloat.c, 183
timlUtilVectorResetInt
util, 98
timlUtilVectorResetInt.c, 183
timlUtilVectorSortFloat
util, 98
timlUtilVectorSortFloat.c, 183
timlUtilVectorSortIndexFloat
util, 98
timlUtilVectorSortIndexFloat.c, 183
timlUtilVectorSumFloat
util, 99
timlUtilVectorSumFloat.c, 184

util, 78
timlUtilAllocatorLevel, 81
timlUtilClassifyAccuracy, 82
timlUtilConv2Full, 82
timlUtilConv2ImageReshape, 83
timlUtilConv2ImageReshapeBack, 83
timlUtilConv2ImageReshapeIndex, 83
timlUtilConv2Valid, 84
timlUtilCorr2Full, 84
timlUtilCropType, 81
timlUtilDiffTime, 84
timlUtilElementWiseFunction, 86
timlUtilElementWiseMultiply, 86
timlUtilFread, 86
timlUtilFree, 86
timlUtilFwrite, 87
timlUtilLocalContrastNormalize, 87
timlUtilLocalContrastUnnormalize, 87
timlUtilMalloc, 88
timlUtilMasking, 88
timlUtilMaxPooling, 88
timlUtilMeanPooling, 89
timlUtilMeanSqaureError, 89
timlUtilMirrorType, 82
timlUtilMultinomialCrossEntropy, 90
timlUtilParamsLevel, 82
timlUtilRandContinuousUniformRNG, 90
timlUtilRandDiscreteUniformRNG, 90
timlUtilRandNormalRNG, 90
timlUtilRandPerm, 91
timlUtilReadCIFAR10, 91
timlUtilReadFixedSizeJPEG, 91
timlUtilReadJPEG, 91
timlUtilReadMNIST, 92
timlUtilRelu, 92
timlUtilReluDerivative, 92
timlUtilReverseEndian32, 92
timlUtilScanJPEG, 93
timlUtilSigmoid, 93
timlUtilSigmoidDerivative, 93
timlUtilSoftmax, 93
timlUtilSubtract, 94
timlUtilTanh, 94
timlUtilTanhDerivative, 94
timlUtilTransform, 95
timlUtilUndoMaxPooling, 95
timlUtilUndoMeanPooling, 95
timlUtilUnmasking, 96
timlUtilVectorMaxFloat, 96
timlUtilVectorMaxIndexFloat, 96
timlUtilVectorResetFloat, 98
timlUtilVectorResetInt, 98
timlUtilVectorSortFloat, 98
timlUtilVectorSortIndexFloat, 98
timlUtilVectorSumFloat, 99
Util_AllocatorLevel1, 81
Util_AllocatorLevel2, 81
Util_AllocatorLevel3, 81
Util_CenterCrop, 82
Util_Mirror, 82
Util_NoMirror, 82
Util_ParamsLevel1, 82
Util_ParamsLevel2, 82
Util_ParamsLevel3, 82
Util_RandomCrop, 82
Util_RandomMirror, 82
Util_AllocatorLevel1
util, 81
Util_AllocatorLevel2
util, 81
Util_AllocatorLevel3
util, 81
Util_CenterCrop
util, 82
Util_Mirror
util, 82
Util_NoMirror
util, 82
Util_ParamsLevel1
util, 82
Util_ParamsLevel2
util, 82
Util_ParamsLevel3
util, 82
Util_RandomCrop
util, 82
Util_RandomMirror
util, 82

val
timlUtilInitializer, 118