

ceras

Generated by Doxygen 1.9.1



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 ceras Namespace Reference	9
5.1.1 Typedef Documentation	18
5.1.1.1 ada_delta	18
5.1.1.2 ada_grad	18
5.1.1.3 cube	19
5.1.1.4 default_allocator	19
5.1.1.5 matrix	19
5.1.1.6 rms_prop	19
5.1.1.7 tesseract	19
5.1.2 Function Documentation	19
5.1.2.1 abs() [1/2]	19
5.1.2.2 abs() [2/2]	20
5.1.2.3 abs_loss()	20
5.1.2.4 Add()	20
5.1.2.5 add()	20
5.1.2.6 amax()	21
5.1.2.7 amin()	21
5.1.2.8 arg()	21
5.1.2.9 as_tensor()	21
5.1.2.10 AveragePooling2D()	21
5.1.2.11 BatchNormalization() [1/2]	22
5.1.2.12 BatchNormalization() [2/2]	22
5.1.2.13 binary_cross_entropy_loss()	22
5.1.2.14 clip()	23
5.1.2.15 computation_graph()	23
5.1.2.16 concatenate()	23
5.1.2.17 Concatenate()	23
5.1.2.18 conj()	24
5.1.2.19 Conv2D()	24
5.1.2.20 copy()	25
5.1.2.21 crelu()	25

5.1.2.22 cross_entropy()	25
5.1.2.23 cross_entropy_loss()	26
5.1.2.24 deep_copy()	26
5.1.2.25 Dense()	26
5.1.2.26 Dropout()	26
5.1.2.27 elementwise_divide()	27
5.1.2.28 elementwise_multiply()	27
5.1.2.29 elementwise_product() [1/2]	27
5.1.2.30 elementwise_product() [2/2]	27
5.1.2.31 elu()	27
5.1.2.32 ELU()	28
5.1.2.33 empty()	28
5.1.2.34 exponential()	28
5.1.2.35 Flatten()	28
5.1.2.36 gelu()	28
5.1.2.37 gemm() [1/2]	28
5.1.2.38 gemm() [2/2]	29
5.1.2.39 gemm_cpu()	29
5.1.2.40 get_default_session()	29
5.1.2.41 glorot_uniform()	29
5.1.2.42 hadamard_product() [1/2]	29
5.1.2.43 hadamard_product() [2/2]	30
5.1.2.44 hard_sigmoid()	30
5.1.2.45 has_inf()	30
5.1.2.46 has_nan()	30
5.1.2.47 hinge_loss()	30
5.1.2.48 imag()	30
5.1.2.49 Input()	31
5.1.2.50 is_valid()	31
5.1.2.51 leaky_relu()	31
5.1.2.52 LeakyReLU()	31
5.1.2.53 linspace()	31
5.1.2.54 load_tensor()	32
5.1.2.55 mae()	32
5.1.2.56 make_compiled_model()	32
5.1.2.57 make_trainable()	32
5.1.2.58 max() [1/2]	32
5.1.2.59 max() [2/2]	33
5.1.2.60 MaxPooling2D()	33
5.1.2.61 mean() [1/3]	33
5.1.2.62 mean() [2/3]	33
5.1.2.63 mean() [3/3]	33

5.1.2.64 mean_absolute_error()	33
5.1.2.65 mean_reduce()	33
5.1.2.66 mean_squared_error()	34
5.1.2.67 mean_squared_logarithmic_error()	34
5.1.2.68 min() [1/2]	34
5.1.2.69 min() [2/2]	34
5.1.2.70 minus() [1/2]	35
5.1.2.71 minus() [2/2]	35
5.1.2.72 mse()	35
5.1.2.73 Multiply()	35
5.1.2.74 multiply() [1/2]	35
5.1.2.75 multiply() [2/2]	36
5.1.2.76 negative()	36
5.1.2.77 negative_relu()	36
5.1.2.78 norm() [1/2]	36
5.1.2.79 norm() [2/2]	36
5.1.2.80 ones()	37
5.1.2.81 ones_like()	37
5.1.2.82 operator!=()	37
5.1.2.83 operator*() [1/7]	37
5.1.2.84 operator*() [2/7]	37
5.1.2.85 operator*() [3/7]	38
5.1.2.86 operator*() [4/7]	38
5.1.2.87 operator*() [5/7]	38
5.1.2.88 operator*() [6/7]	38
5.1.2.89 operator*() [7/7]	38
5.1.2.90 operator+() [1/9]	39
5.1.2.91 operator+() [2/9]	39
5.1.2.92 operator+() [3/9]	39
5.1.2.93 operator+() [4/9]	39
5.1.2.94 operator+() [5/9]	39
5.1.2.95 operator+() [6/9]	40
5.1.2.96 operator+() [7/9]	40
5.1.2.97 operator+() [8/9]	40
5.1.2.98 operator+() [9/9]	40
5.1.2.99 operator-() [1/9]	40
5.1.2.100 operator-() [2/9]	41
5.1.2.101 operator-() [3/9]	41
5.1.2.102 operator-() [4/9]	41
5.1.2.103 operator-() [5/9]	41
5.1.2.104 operator-() [6/9]	41
5.1.2.105 operator-() [7/9]	42

5.1.2.106 operator-() [8/9]	42
5.1.2.107 operator-() [9/9]	42
5.1.2.108 operator/()	42
5.1.2.109 operator<()	42
5.1.2.110 operator<<()	42
5.1.2.111 operator<=()	43
5.1.2.112 operator==(1/2)	43
5.1.2.113 operator==(2/2)	43
5.1.2.114 operator>()	43
5.1.2.115 operator>=()	43
5.1.2.116 plus()	43
5.1.2.117 polar()	44
5.1.2.118 randn()	44
5.1.2.119 randn_like()	44
5.1.2.120 random()	44
5.1.2.121 random_like()	45
5.1.2.122 read_tensor()	45
5.1.2.123 real()	45
5.1.2.124 reduce()	45
5.1.2.125 reduce_mean(1/2)	45
5.1.2.126 reduce_mean(2/2)	46
5.1.2.127 reduce_sum(1/2)	46
5.1.2.128 reduce_sum(2/2)	46
5.1.2.129 relu()	46
5.1.2.130 ReLU()	46
5.1.2.131 relu6()	46
5.1.2.132 repeat()	47
5.1.2.133 replace_placeholder_with_expression()	47
5.1.2.134 repmat()	47
5.1.2.135 Reshape()	47
5.1.2.136 reshape()	48
5.1.2.137 save_tensor()	48
5.1.2.138 selu()	48
5.1.2.139 sigmoid()	48
5.1.2.140 silu()	48
5.1.2.141 Softmax()	48
5.1.2.142 softmax(1/2)	49
5.1.2.143 softmax(2/2)	49
5.1.2.144 softplus()	49
5.1.2.145 softsign()	49
5.1.2.146 square()	49
5.1.2.147 squared_loss()	50

5.1.2.148 squeeze()	50
5.1.2.149 standard_deviation()	50
5.1.2.150 std()	50
5.1.2.151 Subtract()	50
5.1.2.152 sum() [1/2]	51
5.1.2.153 sum() [2/2]	51
5.1.2.154 sum_reduce()	51
5.1.2.155 swish()	51
5.1.2.156 truncated_normal()	51
5.1.2.157 update_cuda_gemm_threshold()	52
5.1.2.158 UpSampling2D()	52
5.1.2.159 var()	52
5.1.2.160 variance()	52
5.1.2.161 write_tensor()	52
5.1.2.162 zeros()	53
5.1.2.163 zeros_like()	53
5.1.3 Variable Documentation	53
5.1.3.1 Adadelta	53
5.1.3.2 Adagrad	53
5.1.3.3 Adam	53
5.1.3.4 Binary_Operator	54
5.1.3.5 BinaryCrossentropy	54
5.1.3.6 BinaryCrossEntropy	54
5.1.3.7 CategoricalCrossentropy	54
5.1.3.8 CategoricalCrossEntropy	55
5.1.3.9 Complex	55
5.1.3.10 Constant	55
5.1.3.11 Expression	55
5.1.3.12 Hinge	55
5.1.3.13 is_binary_operator_v	56
5.1.3.14 is_complex_v	56
5.1.3.15 is_constant_v	56
5.1.3.16 is_place_holder_v	56
5.1.3.17 is_tensor_v	56
5.1.3.18 is_unary_operator_v	56
5.1.3.19 is_value_v	56
5.1.3.20 is_variable_v	57
5.1.3.21 MAE	57
5.1.3.22 make_binary_operator	57
5.1.3.23 make_unary_operator	57
5.1.3.24 MeanAbsoluteError	58
5.1.3.25 MeanSquaredError	58

5.1.3.26 MSE	58
5.1.3.27 Operator	59
5.1.3.28 Place_Holder	59
5.1.3.29 random_generator	59
5.1.3.30 random_seed	59
5.1.3.31 RMSprop	59
5.1.3.32 SGD	59
5.1.3.33 Tensor	60
5.1.3.34 Unary_Operator	60
5.1.3.35 Value	60
5.1.3.36 Variable	60
5.2 ceras::ceras_private Namespace Reference	60
5.3 ceras::dataset Namespace Reference	60
5.4 ceras::dataset::fashion_mnist Namespace Reference	60
5.4.1 Function Documentation	61
5.4.1.1 load_data()	61
5.5 ceras::dataset::mnist Namespace Reference	61
5.5.1 Function Documentation	61
5.5.1.1 load_data()	61
<b>6 Class Documentation</b>	<b>63</b>
6.1 ceras::adadelat< Loss, T > Struct Template Reference	63
6.1.1 Member Typedef Documentation	63
6.1.1.1 tensor_type	64
6.1.2 Constructor & Destructor Documentation	64
6.1.2.1 adadelat()	64
6.1.3 Member Function Documentation	64
6.1.3.1 forward()	64
6.1.4 Member Data Documentation	64
6.1.4.1 iterations_	64
6.1.4.2 learning_rate_	64
6.1.4.3 loss_	65
6.1.4.4 rho_	65
6.2 ceras::adagrad< Loss, T > Struct Template Reference	65
6.2.1 Member Typedef Documentation	65
6.2.1.1 tensor_type	66
6.2.2 Constructor & Destructor Documentation	66
6.2.2.1 adagrad()	66
6.2.3 Member Function Documentation	66
6.2.3.1 forward()	66
6.2.4 Member Data Documentation	66
6.2.4.1 decay_	66



6.2.4.2 iterations_ . . . . .	66
6.2.4.3 learning_rate_ . . . . .	67
6.2.4.4 loss_ . . . . .	67
6.3 ceras::adam< Loss, T > Struct Template Reference . . . . .	67
6.3.1 Member Typedef Documentation . . . . .	68
6.3.1.1 tensor_type . . . . .	68
6.3.2 Constructor & Destructor Documentation . . . . .	68
6.3.2.1 adam() . . . . .	68
6.3.3 Member Function Documentation . . . . .	68
6.3.3.1 forward() . . . . .	68
6.3.4 Member Data Documentation . . . . .	68
6.3.4.1 amsgrad_ . . . . .	68
6.3.4.2 beta_1_ . . . . .	69
6.3.4.3 beta_2_ . . . . .	69
6.3.4.4 iterations_ . . . . .	69
6.3.4.5 learning_rate_ . . . . .	69
6.3.4.6 loss_ . . . . .	69
6.4 ceras::binary_operator< Lhs_Operator, Rh_Operator, Forward_Action, Backward_Action > Struct Template Reference . . . . .	69
6.4.1 Member Typedef Documentation . . . . .	70
6.4.1.1 tensor_type . . . . .	70
6.4.2 Constructor & Destructor Documentation . . . . .	70
6.4.2.1 binary_operator() . . . . .	70
6.4.3 Member Function Documentation . . . . .	71
6.4.3.1 backward() . . . . .	71
6.4.3.2 forward() . . . . .	71
6.4.4 Member Data Documentation . . . . .	71
6.4.4.1 backward_action_ . . . . .	71
6.4.4.2 forward_action_ . . . . .	71
6.4.4.3 lhs_input_data_ . . . . .	71
6.4.4.4 lhs_op_ . . . . .	72
6.4.4.5 output_data_ . . . . .	72
6.4.4.6 rhs_input_data_ . . . . .	72
6.4.4.7 rhs_op_ . . . . .	72
6.5 ceras::compiled_model< Model, Optimizer, Loss > Struct Template Reference . . . . .	72
6.5.1 Member Typedef Documentation . . . . .	73
6.5.1.1 io_layer_type . . . . .	73
6.5.2 Constructor & Destructor Documentation . . . . .	73
6.5.2.1 compiled_model() . . . . .	73
6.5.3 Member Function Documentation . . . . .	74
6.5.3.1 evaluate() . . . . .	74
6.5.3.2 fit() . . . . .	74

6.5.3.3 operator()	75
6.5.3.4 predict()	75
6.5.3.5 train_on_batch()	75
6.5.3.6 trainable()	76
6.5.4 Member Data Documentation	76
6.5.4.1 compiled_optimizer_	76
6.5.4.2 ground_truth_place_holder_	76
6.5.4.3 input_place_holder_	76
6.5.4.4 loss_	76
6.5.4.5 model_	77
6.5.4.6 optimizer_	77
6.5.4.7 optimizer_type	77
6.6 ceras::complex< Real_Ex, Imag_Ex > Struct Template Reference	77
6.6.1 Member Data Documentation	77
6.6.1.1 imag_	77
6.6.1.2 real_	78
6.7 ceras::constant< Tsor > Struct Template Reference	78
6.7.1 Detailed Description	78
6.7.2 Constructor & Destructor Documentation	78
6.7.2.1 constant()	79
6.7.3 Member Function Documentation	79
6.7.3.1 backward()	79
6.7.3.2 forward()	79
6.7.3.3 shape()	79
6.7.4 Member Data Documentation	79
6.7.4.1 data_	79
6.8 ceras::gradient_descent< Loss, T > Struct Template Reference	80
6.8.1 Member Typedef Documentation	80
6.8.1.1 tensor_type	80
6.8.2 Constructor & Destructor Documentation	80
6.8.2.1 gradient_descent()	80
6.8.3 Member Function Documentation	81
6.8.3.1 forward()	81
6.8.4 Member Data Documentation	81
6.8.4.1 learning_rate_	81
6.8.4.2 loss_	81
6.8.4.3 momentum_	81
6.9 ceras::is_binary_operator< T > Struct Template Reference	81
6.10 ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > > Struct Template Reference	82
6.11 ceras::is_complex< T > Struct Template Reference	82
6.12 ceras::is_complex< complex< Real_Ex, Imag_Ex > > Struct Template Reference	82

6.13 <code>ceras::is_constant&lt; T &gt;</code> Struct Template Reference	83
6.14 <code>ceras::is_constant&lt; constant&lt; Tsor &gt; &gt;</code> Struct Template Reference	83
6.15 <code>ceras::is_place_holder&lt; T &gt;</code> Struct Template Reference	83
6.16 <code>ceras::is_place_holder&lt; place_holder&lt; Tsor &gt; &gt;</code> Struct Template Reference	84
6.17 <code>ceras::is_tensor&lt; T &gt;</code> Struct Template Reference	84
6.18 <code>ceras::is_tensor&lt; tensor&lt; T, A &gt; &gt;</code> Struct Template Reference	84
6.19 <code>ceras::is_unary_operator&lt; T &gt;</code> Struct Template Reference	85
6.20 <code>ceras::is_unary_operator&lt; unary_operator&lt; Operator, Forward_Action, Backward_Action &gt; &gt;</code> Struct Template Reference	85
6.21 <code>ceras::is_value&lt; T &gt;</code> Struct Template Reference	85
6.22 <code>ceras::is_value&lt; value&lt; T &gt; &gt;</code> Struct Template Reference	86
6.23 <code>ceras::is_variable&lt; T &gt;</code> Struct Template Reference	86
6.24 <code>ceras::is_variable&lt; variable&lt; Tsor &gt; &gt;</code> Struct Template Reference	86
6.25 <code>ceras::model&lt; Ex, Ph &gt;</code> Struct Template Reference	87
6.25.1 Detailed Description	87
6.25.2 Member Typedef Documentation	87
6.25.2.1 <code>input_layer_type</code>	88
6.25.2.2 <code>output_layer_type</code>	88
6.25.3 Constructor & Destructor Documentation	88
6.25.3.1 <code>model()</code>	88
6.25.4 Member Function Documentation	88
6.25.4.1 <code>compile()</code>	88
6.25.4.2 <code>input()</code>	89
6.25.4.3 <code>load_weights()</code>	89
6.25.4.4 <code>operator()()</code>	89
6.25.4.5 <code>output()</code>	90
6.25.4.6 <code>predict()</code>	90
6.25.4.7 <code>save_weights()</code>	90
6.25.4.8 <code>summary()</code>	91
6.25.4.9 <code>trainable()</code>	91
6.25.5 Member Data Documentation	91
6.25.5.1 <code>expression_</code>	91
6.25.5.2 <code>place_holder_</code>	91
6.26 <code>ceras::place_holder&lt; Tsor &gt;</code> Struct Template Reference	92
6.26.1 Member Typedef Documentation	92
6.26.1.1 <code>tensor_type</code>	92
6.26.2 Constructor & Destructor Documentation	92
6.26.2.1 <code>place_holder()</code> [1/4]	92
6.26.2.2 <code>place_holder()</code> [2/4]	93
6.26.2.3 <code>place_holder()</code> [3/4]	93
6.26.2.4 <code>place_holder()</code> [4/4]	93
6.26.3 Member Function Documentation	93

6.26.3.1 backward()	93
6.26.3.2 bind()	93
6.26.3.3 forward()	93
6.26.3.4 operator=() [1/2]	94
6.26.3.5 operator=() [2/2]	94
6.26.3.6 reset()	94
6.27 ceras::place_holder_state< Tsr > Struct Template Reference	94
6.27.1 Member Data Documentation	94
6.27.1.1 data_	94
6.27.1.2 shape_hint_	95
6.28 ceras::regularizer< Float > Struct Template Reference	95
6.28.1 Member Typedef Documentation	95
6.28.1.1 value_type	95
6.28.2 Constructor & Destructor Documentation	95
6.28.2.1 regularizer()	96
6.28.3 Member Data Documentation	96
6.28.3.1 l1_	96
6.28.3.2 l2_	96
6.28.3.3 synchronized_	96
6.29 ceras::rmsprop< Loss, T > Struct Template Reference	96
6.29.1 Member Typedef Documentation	97
6.29.1.1 tensor_type	97
6.29.2 Constructor & Destructor Documentation	97
6.29.2.1 rmsprop()	97
6.29.3 Member Function Documentation	97
6.29.3.1 forward()	98
6.29.4 Member Data Documentation	98
6.29.4.1 decay_	98
6.29.4.2 iterations_	98
6.29.4.3 learning_rate_	98
6.29.4.4 loss_	98
6.29.4.5 rho_	98
6.30 ceras::ceras_private::session< Tsr > Struct Template Reference	99
6.30.1 Member Typedef Documentation	99
6.30.1.1 place_holder_type	99
6.30.1.2 variable_state_type	100
6.30.1.3 variable_type	100
6.30.2 Constructor & Destructor Documentation	100
6.30.2.1 session() [1/3]	100
6.30.2.2 session() [2/3]	100
6.30.2.3 session() [3/3]	100
6.30.2.4 ~session()	100

6.30.3 Member Function Documentation	101
6.30.3.1 bind()	101
6.30.3.2 deserialize()	101
6.30.3.3 operator=() [1/2]	101
6.30.3.4 operator=() [2/2]	101
6.30.3.5 rebind()	101
6.30.3.6 remember()	102
6.30.3.7 restore()	102
6.30.3.8 run()	102
6.30.3.9 save()	102
6.30.3.10 serialize()	102
6.30.3.11 tap()	102
6.30.4 Member Data Documentation	103
6.30.4.1 place_holders_	103
6.30.4.2 variables_	103
6.31 ceras::sgd< Loss, T > Struct Template Reference	103
6.31.1 Member Typedef Documentation	104
6.31.1.1 tensor_type	104
6.31.2 Constructor & Destructor Documentation	104
6.31.2.1 sgd()	104
6.31.3 Member Function Documentation	104
6.31.3.1 forward()	104
6.31.4 Member Data Documentation	104
6.31.4.1 decay_	105
6.31.4.2 iterations_	105
6.31.4.3 learning_rate_	105
6.31.4.4 loss_	105
6.31.4.5 momentum_	105
6.31.4.6 nesterov_	105
6.32 ceras::tensor< T, Allocator > Struct Template Reference	106
6.32.1 Member Typedef Documentation	107
6.32.1.1 allocator	107
6.32.1.2 self_type	107
6.32.1.3 shared_vector	107
6.32.1.4 value_type	108
6.32.1.5 vector_type	108
6.32.2 Constructor & Destructor Documentation	108
6.32.2.1 tensor() [1/7]	108
6.32.2.2 tensor() [2/7]	108
6.32.2.3 tensor() [3/7]	108
6.32.2.4 tensor() [4/7]	108
6.32.2.5 tensor() [5/7]	109

6.32.2.6 <code>tensor()</code> [6/7]	109
6.32.2.7 <code>tensor()</code> [7/7]	109
6.32.3 Member Function Documentation	109
6.32.3.1 <code>as_scalar()</code>	109
6.32.3.2 <code>as_type()</code>	109
6.32.3.3 <code>begin()</code> [1/2]	109
6.32.3.4 <code>begin()</code> [2/2]	110
6.32.3.5 <code>cbegin()</code>	110
6.32.3.6 <code>cend()</code>	110
6.32.3.7 <code>copy()</code>	110
6.32.3.8 <code>creep_to()</code>	110
6.32.3.9 <code>data()</code> [1/2]	110
6.32.3.10 <code>data()</code> [2/2]	110
6.32.3.11 <code>deep_copy()</code> [1/2]	111
6.32.3.12 <code>deep_copy()</code> [2/2]	111
6.32.3.13 <code>empty()</code>	111
6.32.3.14 <code>end()</code> [1/2]	111
6.32.3.15 <code>end()</code> [2/2]	111
6.32.3.16 <code>map()</code>	111
6.32.3.17 <code>ndim()</code>	112
6.32.3.18 <code>operator*=( )</code> [1/2]	112
6.32.3.19 <code>operator*=( )</code> [2/2]	112
6.32.3.20 <code>operator+=( )</code> [1/2]	112
6.32.3.21 <code>operator+=( )</code> [2/2]	112
6.32.3.22 <code>operator-( )</code>	112
6.32.3.23 <code>operator-=( )</code> [1/2]	113
6.32.3.24 <code>operator-=( )</code> [2/2]	113
6.32.3.25 <code>operator/=( )</code> [1/2]	113
6.32.3.26 <code>operator/=( )</code> [2/2]	113
6.32.3.27 <code>operator=( )</code> [1/2]	113
6.32.3.28 <code>operator=( )</code> [2/2]	113
6.32.3.29 <code>operator[]()</code> [1/2]	114
6.32.3.30 <code>operator[]()</code> [2/2]	114
6.32.3.31 <code>reset()</code>	114
6.32.3.32 <code>reshape()</code>	114
6.32.3.33 <code>resize()</code>	114
6.32.3.34 <code>shape()</code>	115
6.32.3.35 <code>shrink_to()</code>	115
6.32.3.36 <code>size()</code>	115
6.32.3.37 <code>slice()</code>	115
6.32.4 Member Data Documentation	115
6.32.4.1 <code>memory_offset_</code>	115

6.32.4.2 shape_ . . . . .	115
6.32.4.3 vector_ . . . . .	116
6.33 ceras::tensor_deduction< L, R > Struct Template Reference . . . . .	116
6.33.1 Member Typedef Documentation . . . . .	116
6.33.1.1 op_type . . . . .	116
6.33.1.2 tensor_type . . . . .	116
6.34 ceras::unary_operator< Operator, Forward_Action, Backward_Action > Struct Template Reference . . . . .	116
6.34.1 Constructor & Destructor Documentation . . . . .	117
6.34.1.1 unary_operator() . . . . .	117
6.34.2 Member Function Documentation . . . . .	117
6.34.2.1 backward() . . . . .	117
6.34.2.2 forward() . . . . .	117
6.34.3 Member Data Documentation . . . . .	118
6.34.3.1 backward_action_ . . . . .	118
6.34.3.2 forward_action_ . . . . .	118
6.34.3.3 input_data_ . . . . .	118
6.34.3.4 op_ . . . . .	118
6.34.3.5 output_data_ . . . . .	118
6.34.3.6 tensor_type . . . . .	118
6.35 ceras::value< T > Struct Template Reference . . . . .	119
6.35.1 Member Typedef Documentation . . . . .	119
6.35.1.1 value_type . . . . .	119
6.35.2 Constructor & Destructor Documentation . . . . .	119
6.35.2.1 value() [1/4] . . . . .	120
6.35.2.2 value() [2/4] . . . . .	120
6.35.2.3 value() [3/4] . . . . .	120
6.35.2.4 value() [4/4] . . . . .	120
6.35.3 Member Function Documentation . . . . .	120
6.35.3.1 backward() . . . . .	120
6.35.3.2 forward() . . . . .	120
6.35.3.3 operator=() [1/2] . . . . .	121
6.35.3.4 operator=() [2/2] . . . . .	121
6.35.4 Member Data Documentation . . . . .	121
6.35.4.1 data_ . . . . .	121
6.36 ceras::variable< Tsor > Struct Template Reference . . . . .	121
6.36.1 Member Typedef Documentation . . . . .	122
6.36.1.1 tensor_type . . . . .	122
6.36.1.2 value_type . . . . .	122
6.36.2 Constructor & Destructor Documentation . . . . .	122
6.36.2.1 variable() [1/4] . . . . .	123
6.36.2.2 variable() [2/4] . . . . .	123
6.36.2.3 variable() [3/4] . . . . .	123

6.36.2.4 variable() [4/4]	123
6.36.3 Member Function Documentation	123
6.36.3.1 backward()	123
6.36.3.2 contexts() [1/2]	123
6.36.3.3 contexts() [2/2]	124
6.36.3.4 data() [1/2]	124
6.36.3.5 data() [2/2]	124
6.36.3.6 forward()	124
6.36.3.7 gradient() [1/2]	124
6.36.3.8 gradient() [2/2]	124
6.36.3.9 operator=() [1/2]	124
6.36.3.10 operator=() [2/2]	125
6.36.3.11 reset()	125
6.36.3.12 shape()	125
6.36.3.13 trainable() [1/2]	125
6.36.3.14 trainable() [2/2]	125
6.36.4 Member Data Documentation	125
6.36.4.1 regularizer_	125
6.36.4.2 state_	126
6.36.4.3 trainable_	126
6.37 ceras::variable_state< Tsor > Struct Template Reference	126
6.37.1 Member Data Documentation	126
6.37.1.1 contexts_	126
6.37.1.2 data_	126
6.37.1.3 gradient_	127
6.38 ceras::view_2d< T > Struct Template Reference	127
6.38.1 Member Typedef Documentation	128
6.38.1.1 col_type	128
6.38.1.2 const_col_type	128
6.38.1.3 const_row_type	128
6.38.1.4 row_type	128
6.38.1.5 value_type	128
6.38.2 Constructor & Destructor Documentation	128
6.38.2.1 view_2d() [1/3]	129
6.38.2.2 view_2d() [2/3]	129
6.38.2.3 view_2d() [3/3]	129
6.38.3 Member Function Documentation	129
6.38.3.1 begin()	129
6.38.3.2 col()	129
6.38.3.3 col_begin() [1/2]	130
6.38.3.4 col_begin() [2/2]	130
6.38.3.5 col_end() [1/2]	130



6.38.3.6 col_end() [2/2]	130
6.38.3.7 data() [1/2]	130
6.38.3.8 data() [2/2]	130
6.38.3.9 end()	131
6.38.3.10 operator[]() [1/2]	131
6.38.3.11 operator[]() [2/2]	131
6.38.3.12 row()	131
6.38.3.13 row_begin() [1/2]	131
6.38.3.14 row_begin() [2/2]	131
6.38.3.15 row_end() [1/2]	132
6.38.3.16 row_end() [2/2]	132
6.38.3.17 shape()	132
6.38.3.18 size()	132
6.38.4 Member Data Documentation	132
6.38.4.1 col_	132
6.38.4.2 data_	132
6.38.4.3 row_	133
6.38.4.4 transposed_	133
6.39 ceras::view_3d< T > Struct Template Reference	133
6.39.1 Constructor & Destructor Documentation	133
6.39.1.1 view_3d()	133
6.39.2 Member Function Documentation	134
6.39.2.1 operator[]() [1/2]	134
6.39.2.2 operator[]() [2/2]	134
6.39.3 Member Data Documentation	134
6.39.3.1 channel_	134
6.39.3.2 col_	134
6.39.3.3 data_	134
6.39.3.4 row_	135
6.40 ceras::view_4d< T > Struct Template Reference	135
6.40.1 Detailed Description	135
6.40.2 Constructor & Destructor Documentation	135
6.40.2.1 view_4d()	136
6.40.3 Member Function Documentation	137
6.40.3.1 operator[]() [1/2]	137
6.40.3.2 operator[]() [2/2]	137
6.40.4 Member Data Documentation	138
6.40.4.1 batch_size_	138
6.40.4.2 channel_	138
6.40.4.3 col_	138
6.40.4.4 data_	138
6.40.4.5 row_	138

<b>7 File Documentation</b>	<b>139</b>
7.1 /home/feng/workspace/github.repo/ceras/include/activation.hpp File Reference	139
7.2 /home/feng/workspace/github.repo/ceras/include/ceras.hpp File Reference	140
7.3 /home/feng/workspace/github.repo/ceras/include/complex_operator.hpp File Reference	140
7.4 /home/feng/workspace/github.repo/ceras/include/config.hpp File Reference	142
7.5 /home/feng/workspace/github.repo/ceras/include/constant.hpp File Reference	142
7.6 /home/feng/workspace/github.repo/ceras/include/dataset.hpp File Reference	143
7.7 /home/feng/workspace/github.repo/ceras/include/includes.hpp File Reference	143
7.7.1 Macro Definition Documentation	144
7.7.1.1 STB_IMAGE_IMPLEMENTATION	144
7.7.1.2 STB_IMAGE_RESIZE_IMPLEMENTATION	144
7.7.1.3 STB_IMAGE_WRITE_IMPLEMENTATION	144
7.8 /home/feng/workspace/github.repo/ceras/include/layer.hpp File Reference	144
7.9 /home/feng/workspace/github.repo/ceras/include/loss.hpp File Reference	145
7.10 /home/feng/workspace/github.repo/ceras/include/model.hpp File Reference	147
7.11 /home/feng/workspace/github.repo/ceras/include/operation.hpp File Reference	147
7.11.1 Function Documentation	152
7.11.1.1 abs()	152
7.11.1.2 acos()	153
7.11.1.3 acosh()	153
7.11.1.4 asin()	153
7.11.1.5 asinh()	153
7.11.1.6 atan()	154
7.11.1.7 atan2()	154
7.11.1.8 atanh()	154
7.11.1.9 average_pooling_2d()	154
7.11.1.10 batch_normalization()	154
7.11.1.11 cbrt()	155
7.11.1.12 ceil()	155
7.11.1.13 clip()	155
7.11.1.14 concat() [1/2]	155
7.11.1.15 concat() [2/2]	155
7.11.1.16 concatenate() [1/2]	156
7.11.1.17 concatenate() [2/2]	156
7.11.1.18 conv2d()	156
7.11.1.19 cos()	156
7.11.1.20 cosh()	156
7.11.1.21 drop_out()	157
7.11.1.22 equal()	157
7.11.1.23 erf()	157
7.11.1.24 erfc()	158
7.11.1.25 exp()	158

7.11.1.26 exp2()	158
7.11.1.27 expm1()	158
7.11.1.28 fabs()	159
7.11.1.29 flatten()	159
7.11.1.30 floor()	159
7.11.1.31 hypot()	159
7.11.1.32 identity()	159
7.11.1.33 img2col()	160
7.11.1.34 llrint()	160
7.11.1.35 llround()	160
7.11.1.36 log()	160
7.11.1.37 log10()	161
7.11.1.38 log1p()	161
7.11.1.39 log2()	161
7.11.1.40 lrint()	161
7.11.1.41 lround()	162
7.11.1.42 max_pooling_2d()	162
7.11.1.43 maximum()	162
7.11.1.44 nearbyint()	162
7.11.1.45 normalization_batch()	162
7.11.1.46 ones_like()	163
7.11.1.47 random_normal_like()	163
7.11.1.48 reduce_max()	163
7.11.1.49 reduce_min()	164
7.11.1.50 reduce_sum()	164
7.11.1.51 repeat()	164
7.11.1.52 reshape()	165
7.11.1.53 rint()	165
7.11.1.54 round()	165
7.11.1.55 sign()	165
7.11.1.56 sin()	166
7.11.1.57 sinh()	166
7.11.1.58 sqrt()	166
7.11.1.59 tan()	167
7.11.1.60 tanh()	167
7.11.1.61 transpose()	167
7.11.1.62 trunc()	167
7.11.1.63 up_sampling_2d()	167
7.11.1.64 zero_padding_2d()	167
7.11.1.65 zeros_like()	168
7.11.2 Variable Documentation	168
7.11.2.1 sqr	168

7.11.2.2 y . . . . .	168
7.12 /home/feng/workspace/github.repo/ceras/include/optimizer.hpp File Reference . . . . .	168
7.13 /home/feng/workspace/github.repo/ceras/include/place_holder.hpp File Reference . . . . .	169
7.14 /home/feng/workspace/github.repo/ceras/include/session.hpp File Reference . . . . .	170
7.15 /home/feng/workspace/github.repo/ceras/include/tensor.hpp File Reference . . . . .	171
7.16 /home/feng/workspace/github.repo/ceras/include/value.hpp File Reference . . . . .	174
7.17 /home/feng/workspace/github.repo/ceras/include/variable.hpp File Reference . . . . .	175
<b>Index</b>	<b>177</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">ceras</a>	9
<a href="#">ceras::ceras_private</a>	60
<a href="#">ceras::dataset</a>	60
<a href="#">ceras::dataset::fashion_mnist</a>	60
<a href="#">ceras::dataset::mnist</a>	61



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

ceras::compiled_model< Model, Optimizer, Loss > . . . . .	72
ceras::complex< Real_Ex, Imag_Ex > . . . . .	77
enable_id	
ceras::adadelta< Loss, T > . . . . .	63
ceras::adagrad< Loss, T > . . . . .	65
ceras::adam< Loss, T > . . . . .	67
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > . . . . .	69
ceras::constant< Tsor > . . . . .	78
ceras::gradient_descent< Loss, T > . . . . .	80
ceras::place_holder< Tsor > . . . . .	92
ceras::rmsprop< Loss, T > . . . . .	96
ceras::sgd< Loss, T > . . . . .	103
ceras::tensor< T, Allocator > . . . . .	106
ceras::unary_operator< Operator, Forward_Action, Backward_Action > . . . . .	116
ceras::value< T > . . . . .	119
ceras::variable< Tsor > . . . . .	121
enable_shared	
ceras::adadelta< Loss, T > . . . . .	63
ceras::adagrad< Loss, T > . . . . .	65
ceras::adam< Loss, T > . . . . .	67
ceras::gradient_descent< Loss, T > . . . . .	80
ceras::rmsprop< Loss, T > . . . . .	96
ceras::sgd< Loss, T > . . . . .	103
enable_shared_state	
ceras::place_holder< Tsor > . . . . .	92
std::false_type	
ceras::is_binary_operator< T > . . . . .	81
ceras::is_complex< T > . . . . .	82
ceras::is_constant< T > . . . . .	83
ceras::is_place_holder< T > . . . . .	83
ceras::is_tensor< T > . . . . .	84
ceras::is_unary_operator< T > . . . . .	85
ceras::is_value< T > . . . . .	85
ceras::is_variable< T > . . . . .	86
ceras::model< Ex, Ph > . . . . .	87

ceras::place_holder_state< Tsor > . . . . .	94
ceras::regularizer< Float > . . . . .	95
ceras::regularizer< value_type > . . . . .	95
ceras::ceras_private::session< Tsor > . . . . .	99
ceras::tensor_deduction< L, R > . . . . .	116
ceras::tensor_deduction< Lhs_Operator, Rhc_Operator > . . . . .	116
std::true_type	
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > > . . . . .	82
ceras::is_complex< complex< Real_Ex, Imag_Ex > > . . . . .	82
ceras::is_constant< constant< Tsor > > . . . . .	83
ceras::is_place_holder< place_holder< Tsor > > . . . . .	84
ceras::is_tensor< tensor< T, A > > . . . . .	84
ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > > . . . . .	85
ceras::is_value< value< T > > . . . . .	86
ceras::is_variable< variable< Tsor > > . . . . .	86
ceras::variable_state< Tsor > . . . . .	126
ceras::view_2d< T > . . . . .	127
ceras::view_3d< T > . . . . .	133
ceras::view_4d< T > . . . . .	135



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">ceras::adadelta&lt; Loss, T &gt;</a>	63
<a href="#">ceras::adagrad&lt; Loss, T &gt;</a>	65
<a href="#">ceras::adam&lt; Loss, T &gt;</a>	67
<a href="#">ceras::binary_operator&lt; Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action &gt;</a>	69
<a href="#">ceras::compiled_model&lt; Model, Optimizer, Loss &gt;</a>	72
<a href="#">ceras::complex&lt; Real_Ex, Imag_Ex &gt;</a>	77
<a href="#">ceras::constant&lt; Tzor &gt;</a>	
Creates a constant expression from a tensor-like object	78
<a href="#">ceras::gradient_descent&lt; Loss, T &gt;</a>	80
<a href="#">ceras::is_binary_operator&lt; T &gt;</a>	81
<a href="#">ceras::is_binary_operator&lt; binary_operator&lt; Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action &gt; &gt;</a>	82
<a href="#">ceras::is_complex&lt; T &gt;</a>	82
<a href="#">ceras::is_complex&lt; complex&lt; Real_Ex, Imag_Ex &gt; &gt;</a>	82
<a href="#">ceras::is_constant&lt; T &gt;</a>	83
<a href="#">ceras::is_constant&lt; constant&lt; Tzor &gt; &gt;</a>	83
<a href="#">ceras::is_place_holder&lt; T &gt;</a>	83
<a href="#">ceras::is_place_holder&lt; place_holder&lt; Tzor &gt; &gt;</a>	84
<a href="#">ceras::is_tensor&lt; T &gt;</a>	84
<a href="#">ceras::is_tensor&lt; tensor&lt; T, A &gt; &gt;</a>	84
<a href="#">ceras::is_unary_operator&lt; T &gt;</a>	85
<a href="#">ceras::is_unary_operator&lt; unary_operator&lt; Operator, Forward_Action, Backward_Action &gt; &gt;</a>	85
<a href="#">ceras::is_value&lt; T &gt;</a>	85
<a href="#">ceras::is_value&lt; value&lt; T &gt; &gt;</a>	86
<a href="#">ceras::is_variable&lt; T &gt;</a>	86
<a href="#">ceras::is_variable&lt; variable&lt; Tzor &gt; &gt;</a>	86
<a href="#">ceras::model&lt; Ex, Ph &gt;</a>	87
<a href="#">ceras::place_holder&lt; Tzor &gt;</a>	92
<a href="#">ceras::place_holder_state&lt; Tzor &gt;</a>	94
<a href="#">ceras::regularizer&lt; Float &gt;</a>	95
<a href="#">ceras::rmsprop&lt; Loss, T &gt;</a>	96
<a href="#">ceras::ceras_private::session&lt; Tzor &gt;</a>	99
<a href="#">ceras::sgd&lt; Loss, T &gt;</a>	103
<a href="#">ceras::tensor&lt; T, Allocator &gt;</a>	106
<a href="#">ceras::tensor_deduction&lt; L, R &gt;</a>	116

<a href="#">ceras::unary_operator&lt; Operator, Forward_Action, Backward_Action &gt;</a>	116
<a href="#">ceras::value&lt; T &gt;</a>	119
<a href="#">ceras::variable&lt; Tsor &gt;</a>	121
<a href="#">ceras::variable_state&lt; Tsor &gt;</a>	126
<a href="#">ceras::view_2d&lt; T &gt;</a>	127
<a href="#">ceras::view_3d&lt; T &gt;</a>	133
<a href="#">ceras::view_4d&lt; T &gt;</a>	135

## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/home/feng/workspace/github.repo/ceras/include/activation.hpp . . . . .	139
/home/feng/workspace/github.repo/ceras/include/ceras.hpp . . . . .	140
/home/feng/workspace/github.repo/ceras/include/complex_operator.hpp . . . . .	140
/home/feng/workspace/github.repo/ceras/include/config.hpp . . . . .	142
/home/feng/workspace/github.repo/ceras/include/constant.hpp . . . . .	142
/home/feng/workspace/github.repo/ceras/include/dataset.hpp . . . . .	143
/home/feng/workspace/github.repo/ceras/include/includes.hpp . . . . .	143
/home/feng/workspace/github.repo/ceras/include/layer.hpp . . . . .	144
/home/feng/workspace/github.repo/ceras/include/loss.hpp . . . . .	145
/home/feng/workspace/github.repo/ceras/include/model.hpp . . . . .	147
/home/feng/workspace/github.repo/ceras/include/operation.hpp . . . . .	147
/home/feng/workspace/github.repo/ceras/include/optimizer.hpp . . . . .	168
/home/feng/workspace/github.repo/ceras/include/place_holder.hpp . . . . .	169
/home/feng/workspace/github.repo/ceras/include/session.hpp . . . . .	170
/home/feng/workspace/github.repo/ceras/include/tensor.hpp . . . . .	171
/home/feng/workspace/github.repo/ceras/include/value.hpp . . . . .	174
/home/feng/workspace/github.repo/ceras/include/variable.hpp . . . . .	175



## Chapter 5

# Namespace Documentation

### 5.1 ceras Namespace Reference

#### Namespaces

- [ceras\\_private](#)
- [dataset](#)

#### Classes

- struct [complex](#)
- struct [is\\_complex](#)
- struct [is\\_complex< complex< Real\\_Ex, Imag\\_Ex > >](#)
- struct [constant](#)  
*Creates a constant expression from a tensor-like object.*
- struct [is\\_constant](#)
- struct [is\\_constant< constant< Tsor > >](#)
- struct [compiled\\_model](#)
- struct [model](#)
- struct [unary\\_operator](#)
- struct [binary\\_operator](#)
- struct [is\\_unary\\_operator](#)
- struct [is\\_unary\\_operator< unary\\_operator< Operator, Forward\\_Action, Backward\\_Action > >](#)
- struct [is\\_binary\\_operator](#)
- struct [is\\_binary\\_operator< binary\\_operator< Lhs\\_Operator, Rhc\\_Operator, Forward\\_Action, Backward\\_Action > >](#)
- struct [sgd](#)
- struct [adagrad](#)
- struct [rmsprop](#)
- struct [adadelat](#)
- struct [adam](#)
- struct [gradient\\_descent](#)
- struct [place\\_holder\\_state](#)
- struct [place\\_holder](#)
- struct [is\\_place\\_holder](#)
- struct [is\\_place\\_holder< place\\_holder< Tsor > >](#)
- struct [tensor](#)
- struct [is\\_tensor](#)

- struct [is\\_tensor](#)< [tensor](#)< T, A > >
- struct [view\\_2d](#)
- struct [view\\_3d](#)
- struct [view\\_4d](#)
- struct [value](#)
- struct [is\\_value](#)
- struct [is\\_value](#)< [value](#)< T > >
- struct [tensor\\_deduction](#)
- struct [variable\\_state](#)
- struct [regularizer](#)
- struct [variable](#)
- struct [is\\_variable](#)
- struct [is\\_variable](#)< [variable](#)< Tsor > >

## Typedefs

- template<typename Loss , typename T >  
using [ada\\_grad](#) = [adagrad](#)< Loss, T >
- template<typename Loss , typename T >  
using [rms\\_prop](#) = [rmsprop](#)< Loss, T >
- template<typename Loss , typename T >  
using [ada\\_delta](#) = [adadelta](#)< Loss, T >
- template<typename T >  
using [default\\_allocator](#) = std::allocator< T >
- template<typename T >  
using [matrix](#) = [view\\_2d](#)< T >
- template<typename T >  
using [cube](#) = [view\\_3d](#)< T >
- template<typename T >  
using [tesseract](#) = [view\\_4d](#)< T >

## Functions

- template<Expression Ex>  
constexpr auto [softmax](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [selu](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [softplus](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [softsign](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [sigmoid](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [relu](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [relu6](#) (Ex const &ex) noexcept
- template<typename T >  
requires std::floating\_point< T > auto [leaky\\_relu](#) (T const factor) noexcept
- template<Expression Ex>  
auto [negative\\_relu](#) (Ex const &ex) noexcept
- template<typename T = float>  
requires std::floating\_point< T > auto [elu](#) (T const alpha=1.0) noexcept

- template<Expression Ex>  
auto [exponential](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [hard\\_sigmoid](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [gelu](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [swish](#) (Ex const &ex) noexcept  
*Applies the swish activation function.*
- template<Expression Ex>  
auto [silu](#) (Ex const &ex) noexcept  
*An alias name of activation swish.*
- template<Expression Ex>  
auto [crelu](#) (Ex const &ex) noexcept  
*Concatenated Rectified Linear Units, an activation function which preserves both positive and negative phase information while enforcing non-saturated non-linearity.*
- template<Expression Real\_Ex, Expression Imag\_Ex>  
Real\_Ex [real](#) ([complex](#)< Real\_Ex, Imag\_Ex > const &c) noexcept
- template<Expression Real\_Ex, Expression Imag\_Ex>  
Imag\_Ex [imag](#) ([complex](#)< Real\_Ex, Imag\_Ex > const &c) noexcept
- template<Complex C>  
auto [abs](#) (C const &c) noexcept  
*Returns the magnitude of the complex expression.*
- template<Complex C>  
auto [norm](#) (C const &c) noexcept  
*Returns the squared magnitude of the complex expression.*
- template<Complex C>  
auto [conj](#) (C const &c) noexcept  
*Returns the conjugate of the complex expression.*
- template<Expression Em, Expression Ep>  
auto [polar](#) (Em const &em, Ep const &ep) noexcept  
*Returns with given magnitude and phase angle.*
- template<Complex C>  
auto [arg](#) (C const &c) noexcept  
*Calculates the phase angle (in radians) of the complex expression.*
- template<Complex C>  
auto [operator+](#) (C const &c) noexcept  
*Returns the complex expression.*
- template<Complex C>  
auto [operator-](#) (C const &c) noexcept  
*Negatives the complex expression.*
- template<Complex Cl, Complex Cr>  
auto [operator+](#) (Cl const &cl, Cr const &cr) noexcept  
*Sums up two complex expressions.*
- template<Complex Cl, Complex Cr>  
auto [operator-](#) (Cl const &cl, Cr const &cr) noexcept  
*Subtracts one complex expression from the other one.*
- template<Complex Cl, Complex Cr>  
auto [operator\\*](#) (Cl const &cl, Cr const &cr) noexcept  
*Multiplies two complex expressions. Optimization here:  $(a+ib)*(c+id) = (ac-bd) + i(ad+bc) = (ac-bd) + i((a+b)*(c+d)-ac-bd)$*
- template<Complex C, Expression E>  
auto [operator+](#) (C const &c, E const &e) noexcept  
*Sums up a complex expression and an expression.*

- `template<Complex C, Expression E>`  
`auto operator+ (E const &e, C const &c) noexcept`  
*Sums up a complex expression and an expression.*
- `template<Complex C, Expression E>`  
`auto operator- (C const &c, E const &e) noexcept`  
*Subtracts an expression from a compression expression.*
- `template<Complex C, Expression E>`  
`auto operator- (E const &e, C const &c) noexcept`  
*Subtracts a complex expression from an expression.*
- `template<Complex C, Expression E>`  
`auto operator\* (C const &c, E const &e) noexcept`  
*Multiplies a complex expression with an expression.*
- `template<Complex C, Expression E>`  
`auto operator\* (E const &e, C const &c) noexcept`  
*Multiplies an expression with a compression expression.*
- `auto Input ()`
- `auto Conv2D (unsigned long output_channels, std::vector< unsigned long > const &kernel_size, std::vector< unsigned long > const &input_shape, std::string const &padding="valid", std::vector< unsigned long > const &strides={1, 1}, std::vector< unsigned long > const &dilations={1, 1}, bool use_bias=true, float kernel_regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_regularizer_l2=0.0f)`  
*2D convolution layer.*
- `auto Dense (unsigned long output_size, unsigned long input_size, bool use_bias=true, float kernel_regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_regularizer_l2=0.0f)`  
*Densely-connected layer.*
- `auto BatchNormalization (std::vector< unsigned long > const &shape, float threshold=0.95f, float kernel_regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_regularizer_l2=0.0f)`  
*Applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.*
- `auto BatchNormalization (float threshold, std::vector< unsigned long > const &shape, float kernel_regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_regularizer_l2=0.0f)`
- `auto Concatenate (unsigned long axis=-1) noexcept`
- `auto Add () noexcept`
- `auto Subtract () noexcept`
- `auto Multiply () noexcept`
- `template<Expression Ex>`  
`auto ReLU (Ex const &ex) noexcept`
- `auto Softmax () noexcept`
- `template<typename T = float>`  
`auto LeakyReLU (T const factor=0.2) noexcept`
- `template<typename T = float>`  
`auto ELU (T const factor=0.2) noexcept`
- `auto Reshape (std::vector< unsigned long > const &new_shape, bool include_batch_flag=true) noexcept`
- `auto Flatten () noexcept`
- `auto MaxPooling2D (unsigned long stride) noexcept`
- `auto UpSampling2D (unsigned long stride) noexcept`
- `template<typename T >`  
`auto Dropout (T factor) noexcept`
- `auto AveragePooling2D (unsigned long stride) noexcept`
- `template<Expression Lhs_Expression, Expression Rhc_Expression>`  
`constexpr auto mean\_squared\_logarithmic\_error (Lhs_Expression const &lhc_ex, Rhc_Expression const &rhc_ex) noexcept`



- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto squared\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto mean\_squared\_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto mse (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto abs\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto mean\_absolute\_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto mae (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto cross\_entropy (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto binary\_cross\_entropy\_loss (Lhs_Expression const &ground_truth, Rhs_Expression const &prediction) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto cross\_entropy\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto hinge\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`void make\_trainable (Ex &ex, bool t)`
- `template<Expression Ex, Place_Holder Ph, Expression Ey>`  
`auto replace\_placeholder\_with\_expression (Ex const &ex, Ph const &old_place_holder, Ey const &new_↵  
expression)`
- `template<typename Model , typename Optimizer , typename Loss >`  
`auto make\_compiled\_model (Model const &m, Loss const &l, Optimizer const &o)`
- `template<Expression Ex>`  
`std::string computation\_graph (Ex const &ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto plus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto operator+ (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto operator+ (Ex const &ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`auto operator\* (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto negative (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto operator- (Ex const &ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto elementwise\_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto elementwise\_multiply (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto hadamard\_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto sum\_reduce (Ex const &ex) noexcept`

- `template<Expression Ex>`  
`constexpr auto reduce\_sum (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto mean\_reduce (Ex const &ex) noexcept`  
*Computes the mean of elements across all dimensions of an expression.*
- `template<Expression Ex>`  
`constexpr auto reduce\_mean (Ex const &ex) noexcept`  
*An alias name of [mean\\_reduce](#).*
- `template<Expression Ex>`  
`constexpr auto mean (Ex const &ex) noexcept`  
*An alias name of [mean\\_reduce](#).*
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto minus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto operator- (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto square (Ex const &ex) noexcept`
- `template<Place_Holder Ph>`  
`bool operator== (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool operator!= (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool operator< (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool operator> (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool operator<= (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool operator>= (Ph const &lhs, Ph const &rhs)`
- `template<Tensor Tsor>`  
`ceras\_private::session< Tsor > &get\_default\_session ()`
- `template<typename T, typename A = default_allocator<T>>`  
`constexpr tensor< T, A > as\_tensor (T val) noexcept`
- `template<Tensor Tsor, typename CharT, typename Traits >`  
`std::basic_ostream< CharT, Traits > &operator<< (std::basic_ostream< CharT, Traits > &os_, Tsor const &tsor)`
- `template<typename T >`  
`requires std::floating_point< T > void gemm\_cpu (T const *A, bool a_transposed, T const *B, bool b_transposed, unsigned long m, unsigned long n, unsigned long k, T *C)`
- `void update\_cuda\_gemm\_threshold ()`
- `template<typename T >`  
`requires std::floating_point< T > void gemm (T const *A, bool a_transposed, T const *B, bool b_transposed, unsigned long m, unsigned long n, unsigned long k, T *C)`
- `template<typename T >`  
`requires std::floating_point< T > void gemm (view\_2d< T > const &x, view\_2d< T > const &y, view\_2d< T > &ans)`
- `template<Tensor Tsor>`  
`Tsor add (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`  
`Tsor operator+ (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`  
`Tsor operator+ (typename Tsor::value_type const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`  
`Tsor operator+ (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept`
- `template<Tensor Tsor>`  
`Tsor minus (Tsor const &lhs, Tsor const &rhs) noexcept`

- `template<Tensor Tsor>`  
`Tsor operator-` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor operator-` (typename Tsor::value\_type const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor operator-` (Tsor const &lhs, typename Tsor::value\_type const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor operator*` (typename Tsor::value\_type const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor operator*` (Tsor const &lhs, typename Tsor::value\_type const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor operator/` (Tsor const &lhs, typename Tsor::value\_type const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor reshape` (Tsor const &ts, std::vector< unsigned long > const &new\_shape)
- `template<Tensor Tsor>`  
`void multiply` (Tsor const &lhs, Tsor const &rhs, Tsor &ans) noexcept
- `template<Tensor Tsor>`  
`Tsor multiply` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor operator*` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor elementwise_product` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor hadamard_product` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor elementwise_divide` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
`Tsor repeat` (Tsor const &tsor, unsigned long n)
- `template<Tensor Tsor>`  
`Tsor reduce_sum` (Tsor const &tsor)
- `template<Tensor Tsor>`  
`Tsor reduce_mean` (Tsor const &tsor)
- `template<Tensor Tsor>`  
`Tsor clip` (Tsor &tsor, typename Tsor::value\_type lower=0, typename Tsor::value\_type upper=1)
- `template<Tensor Tsor>`  
`Tsor squeeze` (Tsor const &tsor)
- `template<typename T , typename A = default_allocator<T>>>`  
`tensor< T, A > randn` (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1})
- `template<typename T , typename A = default_allocator<T>>>`  
`tensor< T, A > truncated_normal` (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1}, T lower=T{0}, T upper=T{1})
- `template<typename T , typename A = default_allocator<T>>>`  
`tensor< T, A > random` (std::vector< unsigned long > const &shape, T min=T{0}, T max=T{1})
- `template<Tensor Tsor>`  
`Tsor random_like` (Tsor const &tsor, typename Tsor::value\_type min=0, typename Tsor::value\_type max=1)
- `template<Tensor Tsor>`  
`Tsor randn_like` (Tsor const &tsor, typename Tsor::value\_type mean=0, typename Tsor::value\_type stddev=1)
- `template<typename T , typename A = default_allocator<T>>>`  
`tensor< T, A > glorot_uniform` (std::initializer\_list< unsigned long > shape)
- `template<Tensor Tsor>`  
`Tsor deep_copy` (Tsor const &tsor)
- `template<Tensor Tsor>`  
`Tsor copy` (Tsor const &tsor)
- `template<Tensor Tsor>`  
`Tsor concatenate` (Tsor const &lhs, Tsor const &rhs, unsigned long axis=0) noexcept

- `template<Tensor Tsor>`  
`Tsor repmat (Tsor const &tsor, unsigned long row_rep, unsigned long col_rep)`
- `template<Tensor Tsor>`  
`constexpr bool empty (Tsor const &tsor) noexcept`
- `template<typename T, typename A = default_allocator<T>>>`  
`constexpr tensor< T, A > zeros (std::vector< unsigned long > const &shape)`
- `template<Tensor Tsor>`  
`constexpr Tsor zeros\_like (Tsor const &tsor)`
- `template<typename T, typename A = default_allocator<T>>>`  
`constexpr tensor< T, A > ones (std::vector< unsigned long > const &shape)`
- `template<Tensor Tsor>`  
`constexpr Tsor ones\_like (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`auto max (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`auto amax (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`auto min (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`auto amin (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`auto sum (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`auto mean (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`auto norm (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`Tsor abs (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`Tsor softmax (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`bool has\_nan (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`bool has\_inf (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`bool is\_valid (Tsor const &tsor)`
- `template<Tensor Tsor, typename Function >`  
`Tsor reduce (Tsor const &ts, unsigned long axis, typename Tsor::value_type const &init, Function const &func, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`  
`Tsor sum (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`  
`requires std::floating_point< typename Tsor::value_type > Tsor mean (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`  
`requires std::floating_point< typename Tsor::value_type > Tsor variance (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`  
`requires std::floating_point< typename Tsor::value_type > Tsor standard\_deviation (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`  
`requires std::floating_point< typename Tsor::value_type > Tsor::value_type var (Tsor const &ts) noexcept`
- `template<Tensor Tsor>`  
`requires std::floating_point< typename Tsor::value_type > Tsor::value_type std (Tsor const &ts) noexcept`
- `template<Tensor Tsor>`  
`Tsor max (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`

- `template<Tensor Tsor>`  
`Tsor min` (`Tsor const &ts`, unsigned long axis, bool keepdims=false) noexcept
- `template<typename T, typename A = default_allocator<T>>`  
requires `std::floating_point< T >` `tensor< T, A >` `linspace` (`T start`, `T stop`, unsigned long num, bool end-point=true) noexcept
- `template<class _Tp, class _CharT, class _Traits, class _Alloc >`  
`std::basic_istream< _CharT, _Traits > & read_tensor` (`std::basic_istream< _CharT, _Traits > &__is`, `tensor< _Tp, _Alloc > &__x`)
- `template<class _Tp, class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & write_tensor` (`std::basic_ostream< _CharT, _Traits > &__os`, `tensor< _Tp, _Alloc > const &__x`)
- `template<typename T, typename A = default_allocator<T>>`  
`tensor< T, A >` `load_tensor` (`std::string const &file_name`)
- `template<Tensor Tsor>`  
void `save_tensor` (`std::string const &file_name`, `Tsor const &tsor`)
- `template<Variable Var>`  
bool `operator==` (`Var const &lhs`, `Var const &rhs`) noexcept

## Variables

- `template<typename T >`  
constexpr bool `is_complex_v` = `is_complex<T>::value`
- `template<typename T >`  
concept `Complex` = `is_complex_v<T>`  
*A type that represents a complex expression.*
- `template<class T >`  
constexpr bool `is_constant_v` = `is_constant<T>::value`
- `template<typename T >`  
concept `Constant` = `is_constant_v<T>`
- auto `MeanSquaredError`  
*Computes the mean of squares of errors between labels and predictions.*
- auto `MSE`  
*An alias name of function `MeanSquaredError`.*
- auto `MeanAbsoluteError`  
*Computes the mean of absolute errors between labels and predictions.*
- auto `MAE`  
*An alias name of function `MeanAbsoluteError`.*
- auto `Hinge`
- auto `CategoricalCrossentropy`
- auto `CategoricalCrossEntropy`
- auto `BinaryCrossentropy`
- auto `BinaryCrossEntropy`
- static constexpr auto `make_unary_operator`
- static constexpr auto `make_binary_operator`
- `template<class T >`  
constexpr bool `is_unary_operator_v` = `is_unary_operator<T>::value`
- `template<typename T >`  
concept `Unary_Operator` = `is_unary_operator_v<T>`  
*A type that represents an unary operator.*
- `template<class T >`  
constexpr bool `is_binary_operator_v` = `is_binary_operator<T>::value`
- `template<typename T >`  
concept `Binary_Operator` = `is_binary_operator_v<T>`  
*A type that represents a binary operator.*

- `template<typename T>`  
`concept Operator = Unary\_Operator<T> || Binary\_Operator<T>`  
*A type that represents an unary or a binary operator.*
- `template<typename T>`  
`concept Expression = Operator<T> || Variable<T> || Place\_Holder<T> || Constant<T> || Value<T>`  
*A type that represents a unary operator, a binary operator, a variable, a [place\\_holder](#), a constant or a value.*
- `auto Adam`
- `auto SGD`
- `auto Adagrad`
- `auto RMSprop`
- `auto Adadelta`
- `template<class T>`  
`constexpr bool is\_place\_holder\_v = is\_place\_holder<T>::value`
- `template<typename T>`  
`concept Place\_Holder = is\_place\_holder\_v<T>`
- `static unsigned long random\_seed = std::chrono::system_clock::now().time_since_epoch().count()`
- `static std::mt19937 random\_generator {random\_seed}`
- `template<class T>`  
`constexpr bool is\_tensor\_v = is\_tensor<T>::value`
- `template<typename T>`  
`concept Tensor = is\_tensor\_v<T>`
- `template<class T>`  
`constexpr bool is\_value\_v = is\_value<T>::value`
- `template<typename T>`  
`concept Value = is\_value\_v<T>`
- `template<class T>`  
`constexpr bool is\_variable\_v = is\_variable<T>::value`
- `template<typename T>`  
`concept Variable = is\_variable\_v<T>`

## 5.1.1 Typedef Documentation

### 5.1.1.1 `ada_delta`

```
template<typename Loss , typename T >
using ceras::ada\_delta = typedef adadelta< Loss, T >
```

### 5.1.1.2 `ada_grad`

```
template<typename Loss , typename T >
using ceras::ada\_grad = typedef adagrad<Loss, T>
```

### 5.1.1.3 cube

```
template<typename T >
using ceras::cube = typedef view_3d<T>
```

### 5.1.1.4 default\_allocator

```
template<typename T >
using ceras::default_allocator = typedef std::allocator<T>
```

### 5.1.1.5 matrix

```
template<typename T >
using ceras::matrix = typedef view_2d<T>
```

### 5.1.1.6 rms\_prop

```
template<typename Loss , typename T >
using ceras::rms_prop = typedef rmsprop< Loss, T >
```

### 5.1.1.7 tesseract

```
template<typename T >
using ceras::tesseract = typedef view_4d<T>
```

## 5.1.2 Function Documentation

### 5.1.2.1 abs() [1/2]

```
template<Complex C>
auto ceras::abs (
    C const & c ) [noexcept]
```

Returns the magnitude of the complex expression.

## Parameters

<b>c</b>	Complex expression.
----------	---------------------

```
auto r = variable{ ... };
auto i = variable{ ... };
auto c = complex{ r, i };
auto a = abs( c );
```

**5.1.2.2 abs()** [2/2]

```
template<Tensor Tsor>
Tsor ceras::abs (
    Tsor const & tsor )
```

**5.1.2.3 abs\_loss()**

```
template<Expression Lhs_Expression, Expression Rhx_Expression>
constexpr auto ceras::abs_loss (
    Lhs_Expression const & lhs_ex,
    Rhx_Expression const & rhs_ex ) [constexpr], [noexcept]
```

**5.1.2.4 Add()**

```
auto ceras::Add ( ) [inline], [noexcept]
```

Layer that adds two layers

Example usage:

```
auto input = Input(); // (16, )
auto x1 = Dense( 8, 16 )( input );
auto x2 = Dense( 8, 16 )( input );
auto x3 = Add()( x1, x2 ); // equivalent to 'x1 + x2'
auto m = model{ input, x3 };
```

**5.1.2.5 add()**

```
template<Tensor Tsor>
Tsor ceras::add (
    Tsor const & lhs,
    Tsor const & rhs ) [noexcept]
```



### 5.1.2.6 amax()

```
template<Tensor Tsor>
auto ceras::amax (
    Tsor const & tsor )
```

### 5.1.2.7 amin()

```
template<Tensor Tsor>
auto ceras::amin (
    Tsor const & tsor )
```

### 5.1.2.8 arg()

```
template<Complex C>
auto ceras::arg (
    C const & c ) [noexcept]
```

Calculates the phase angle (in radians) of the complex expression.

#### Parameters

<i>c</i>	Complex expression. Implemented as <code>atan2 ( imagec), real(c) ).</code>
----------	---

```
auto r = variable{ ... };
auto i = variable{ ... };
auto c = complex{ r, i };
auto a = arg( c );
```

### 5.1.2.9 as\_tensor()

```
template<typename T , typename A = default_allocator<T>>
constexpr tensor<T, A> ceras::as_tensor (
    T val ) [constexpr], [noexcept]
```

### 5.1.2.10 AveragePooling2D()

```
auto ceras::AveragePooling2D (
    unsigned long stride ) [inline], [noexcept]
```

Average pooling operation for spatial data.

### 5.1.2.11 BatchNormalization() [1/2]

```
auto ceras::BatchNormalization (
    float threshold,
    std::vector< unsigned long > const & shape,
    float kernel_regularizer_l1 = 0.0f,
    float kernel_regularizer_l2 = 0.0f,
    float bias_regularizer_l1 = 0.0f,
    float bias_regularizer_l2 = 0.0f ) [inline]
```

### 5.1.2.12 BatchNormalization() [2/2]

```
auto ceras::BatchNormalization (
    std::vector< unsigned long > const & shape,
    float threshold = 0.95f,
    float kernel_regularizer_l1 = 0.0f,
    float kernel_regularizer_l2 = 0.0f,
    float bias_regularizer_l1 = 0.0f,
    float bias_regularizer_l2 = 0.0f ) [inline]
```

Applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

#### Parameters

<i>shape</i>	Dimensionality of the input shape.
<i>threshold</i>	Momentum for the moving average.
<i>kernel_regularizer_l1</i>	L1 regularizer for the kernel. Defaults to 0.0f.
<i>kernel_regularizer_l2</i>	L2 regularizer for the kernel. Defaults to 0.0f.
<i>bias_regularizer_l1</i>	L1 regularizer for the bias vector. Defaults to 0.0f.
<i>bias_regularizer_l2</i>	L2 regularizer for the bias vector. Defaults to 0.0f.

#### Example code:

```
auto a = variable{ random<float>( {12, 34, 56, 78} ) };
auto b = BatchNormalization( {34, 56, 78}, 0.8f )( a ); // note the leading dimension of 'a' is interpreted
               as batch size, and only the rest 3 dimensions are required here.
```

### 5.1.2.13 binary\_cross\_entropy\_loss()

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::binary_cross_entropy_loss (
    Lhs_Expression const & ground_truth,
    Rhs_Expression const & prediction ) [constexpr], [noexcept]
```

#### 5.1.2.14 clip()

```
template<Tensor Tsor>
Tsor ceras::clip (
    Tsor & tsor,
    typename Tsor::value_type lower = 0,
    typename Tsor::value_type upper = 1 )
```

#### 5.1.2.15 computation\_graph()

```
template<Expression Ex>
std::string ceras::computation_graph (
    Ex const & ex ) [inline], [noexcept]
```

Generating the computation graph, in [graph description language](#).

##### Parameters

<i>ex</i>	An expression.
-----------	----------------

##### Returns

A string describing the computation graph, in graph description language.

#### 5.1.2.16 concatenate()

```
template<Tensor Tsor>
Tsor ceras::concatenate (
    Tsor const & lhs,
    Tsor const & rhs,
    unsigned long axis = 0 ) [noexcept]
```

#### 5.1.2.17 Concatenate()

```
auto ceras::Concatenate (
    unsigned long axis = -1 ) [inline], [noexcept]
```

Layer that concatenates two layers.

##### Parameters

<i>axis</i>	The concatenation axis. Default to the last channel.
-------------	--

Example usage:

```
auto l1 = variable{ tensor<float>{ {12, 11, 3} } };
auto l2 = variable{ tensor<float>{ {12, 11, 4} } };
auto l12 = Concatenate()( l1, l2 ); // should be of shape (12, 11, 7)
```

### 5.1.2.18 conj()

```
template<Complex C>
auto ceras::conj (
    C const & c ) [noexcept]
```

Returns the conjugate of the complex expression.

#### Parameters

<i>c</i>	Complex expression.
----------	---------------------

```
auto r = variable{ ... };
auto i = variable{ ... };
auto c = complex{ r, i };
auto a = conj( c );
```

### 5.1.2.19 Conv2D()

```
auto ceras::Conv2D (
    unsigned long output_channels,
    std::vector< unsigned long > const & kernel_size,
    std::vector< unsigned long > const & input_shape,
    std::string const & padding = "valid",
    std::vector< unsigned long > const & strides = {1,1},
    std::vector< unsigned long > const & dilations = {1, 1},
    bool use_bias = true,
    float kernel_regularizer_l1 = 0.0f,
    float kernel_regularizer_l2 = 0.0f,
    float bias_regularizer_l1 = 0.0f,
    float bias_regularizer_l2 = 0.0f ) [inline]
```

2D convolution layer.

#### Parameters

<i>output_channels</i>	Dimensionality of the output space.
<i>kernel_size</i>	The height and width of the convolutional window.
<i>input_shape</i>	Dimensionality of the input shape.
<i>padding</i>	valid or same. valid suggests no padding. same suggests zero padding. Defaults to valid.
<i>strides</i>	The strides along the height and width direction. Defaults to (1, 1).
<i>dilations</i>	The dialation along the height and width direction. Defaults to (1, 1).
<i>use_bias</i>	Wether or not use a bias vector. Defaults to true.
<i>kernel_regularizer_l1</i>	L1 regularizer for the kernel. Defaults to 0.0f.

## Parameters

<i>kernel_regularizer_↵ _l2</i>	L2 regularizer for the kernel. Defaults to 0.0f.
<i>bias_regularizer_l1</i>	L1 regularizer for the bias vector. Defaults to 0.0f.
<i>bias_regularizer_l2</i>	L2 regularizer for the bias vector. Defaults to 0.0f.

## Example code:

```

auto x = Input{};
auto y = Conv2D( 32, {3, 3}, {28, 28, 1}, "same" )( x );
auto z = Flatten()( y );
auto u = Dense( 10, 28*28*32 )( z );
auto m = model{ x, u };

```

## 5.1.2.20 copy()

```

template<Tensor Tsor>
Tsor ceras::copy (
    Tsor const & tsor )

```

## 5.1.2.21 crelu()

```

template<Expression Ex>
auto ceras::crelu (
    Ex const & ex ) [noexcept]

```

Concatenated Rectified Linear Units, an activation function which preserves both positive and negative phase information while enforcing non-saturated non-linearity.

Reference: Shang, Wenling, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. "Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units." ArXiv:1603.05201 [Cs], July 19, 2016.

<http://arxiv.org/abs/1603.05201>.

```

auto v = variable{ random<float>{ 3, 3 } };
auto c = crelu( v );

```

## 5.1.2.22 cross\_entropy()

```

template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::cross_entropy (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]

```

### 5.1.2.23 cross\_entropy\_loss()

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::cross_entropy_loss (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

### 5.1.2.24 deep\_copy()

```
template<Tensor Tsor>
Tsor ceras::deep_copy (
    Tsor const & tsor )
```

### 5.1.2.25 Dense()

```
auto ceras::Dense (
    unsigned long output_size,
    unsigned long input_size,
    bool use_bias = true,
    float kernel_regularizer_l1 = 0.0f,
    float kernel_regularizer_l2 = 0.0f,
    float bias_regularizer_l1 = 0.0f,
    float bias_regularizer_l2 = 0.0f ) [inline]
```

Densely-connected layer.

#### Parameters

<i>output_size</i>	Dimensionality of output shape. The output shape is (batch_size, output_size).
<i>input_size</i>	Dimensionality of input shape. The input shape is (batch_size, input_size).
<i>use_bias</i>	Using a bias vector or not. Defaults to <code>true</code> .
<i>kernel_regularizer_l1</i>	L1 regularizer for the kernel. Defaults to <code>0.0f</code> .
<i>kernel_regularizer_l2</i>	L2 regularizer for the kernel. Defaults to <code>0.0f</code> .
<i>bias_regularizer_l1</i>	L1 regularizer for the bias vector. Defaults to <code>0.0f</code> .
<i>bias_regularizer_l2</i>	L2 regularizer for the bias vector. Defaults to <code>0.0f</code> .

Example code:

```
auto x = Input{};
auto y = Dense( 10, 28*28 )( x );
auto m = model{ x, y };
```

### 5.1.2.26 Dropout()

```
template<typename T >
```

```
auto ceras::Dropout (
    T factor ) [inline], [noexcept]
```

Applies Dropout to the input.

#### 5.1.2.27 `elementwise_divide()`

```
template<Tensor Tsor>
Tsor ceras::elementwise_divide (
    Tsor const & lhs,
    Tsor const & rhs ) [noexcept]
```

#### 5.1.2.28 `elementwise_multiply()`

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::elementwise_multiply (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

#### 5.1.2.29 `elementwise_product()` [1/2]

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::elementwise_product (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

#### 5.1.2.30 `elementwise_product()` [2/2]

```
template<Tensor Tsor>
Tsor ceras::elementwise_product (
    Tsor const & lhs,
    Tsor const & rhs ) [noexcept]
```

#### 5.1.2.31 `elu()`

```
template<typename T = float>
requires std::floating_point<T> auto ceras::elu (
    T const alpha = 1.0 ) [noexcept]
```

### 5.1.2.32 ELU()

```
template<typename T = float>
auto ceras::ELU (
    T const factor = 0.2 ) [inline], [noexcept]
```

Exponential Linear Unit.

### 5.1.2.33 empty()

```
template<Tensor Tsor>
constexpr bool ceras::empty (
    Tsor const & tsor ) [constexpr], [noexcept]
```

### 5.1.2.34 exponential()

```
template<Expression Ex>
auto ceras::exponential (
    Ex const & ex ) [inline], [noexcept]
```

### 5.1.2.35 Flatten()

```
auto ceras::Flatten ( ) [inline], [noexcept]
```

Flattens the input. Does not affect the batch size.

### 5.1.2.36 gelu()

```
template<Expression Ex>
auto ceras::gelu (
    Ex const & ex ) [inline], [noexcept]
```

### 5.1.2.37 gemm() [1/2]

```
template<typename T >
requires std::floating_point<T> void ceras::gemm (
    T const * A,
    bool a_transposed,
    T const * B,
    bool b_transposed,
    unsigned long m,
    unsigned long n,
    unsigned long k,
    T * C )
```



**5.1.2.38 gemm() [2/2]**

```
template<typename T >
requires std::floating_point<T> void ceras::gemm (
    view_2d< T > const & x,
    view_2d< T > const & y,
    view_2d< T > & ans )
```

**5.1.2.39 gemm\_cpu()**

```
template<typename T >
requires std::floating_point<T> void ceras::gemm_cpu (
    T const * A,
    bool a_transposed,
    T const * B,
    bool b_transposed,
    unsigned long m,
    unsigned long n,
    unsigned long k,
    T * C )
```

**5.1.2.40 get\_default\_session()**

```
template<Tensor Tsor>
ceras_private::session< Tsor > & ceras::get_default_session ( )
```

**5.1.2.41 glorot\_uniform()**

```
template<typename T , typename A = default_allocator<T>>
tensor<T,A> ceras::glorot_uniform (
    std::initializer_list< unsigned long > shape )
```

**5.1.2.42 hadamard\_product() [1/2]**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::hadamard_product (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr, [noexcept]]
```

**5.1.2.43 hadamard\_product()** [2/2]

```
template<Tensor Tsor>
Tsor ceras::hadamard_product (
    Tsor const & lhs,
    Tsor const & rhs ) [noexcept]
```

**5.1.2.44 hard\_sigmoid()**

```
template<Expression Ex>
auto ceras::hard_sigmoid (
    Ex const & ex ) [inline], [noexcept]
```

**5.1.2.45 has\_inf()**

```
template<Tensor Tsor>
bool ceras::has_inf (
    Tsor const & tsor )
```

**5.1.2.46 has\_nan()**

```
template<Tensor Tsor>
bool ceras::has_nan (
    Tsor const & tsor )
```

**5.1.2.47 hinge\_loss()**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::hinge_loss (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

**5.1.2.48 imag()**

```
template<Expression Real_Ex, Expression Imag_Ex>
Imag_Ex ceras::imag (
    complex< Real_Ex, Imag_Ex > const & c ) [noexcept]
```

@bref Returns the imaginary part of the complex expression.

## Parameters

<i>c</i>	A complex expression.
----------	-----------------------

**5.1.2.49 Input()**

```
auto ceras::Input ( ) [inline]
```

**5.1.2.50 is\_valid()**

```
template<Tensor Tsor>
bool ceras::is_valid (
    Tsor const & tsor )
```

**5.1.2.51 leaky\_relu()**

```
template<typename T >
requires std::floating_point<T> auto ceras::leaky_relu (
    T const factor ) [noexcept]
```

**5.1.2.52 LeakyReLU()**

```
template<typename T = float>
auto ceras::LeakyReLU (
    T const factor = 0.2 ) [inline], [noexcept]
```

leaky relu activation function.

**5.1.2.53 linspace()**

```
template<typename T , typename A = default_allocator<T>>
requires std::floating_point<T> tensor<T,A> ceras::linspace (
    T start,
    T stop,
    unsigned long num,
    bool endpoint = true ) [noexcept]
```

**5.1.2.54 load\_tensor()**

```
template<typename T , typename A = default_allocator<T>>
tensor<T,A> ceras::load_tensor (
    std::string const & file_name )
```

**5.1.2.55 mae()**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::mae (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

**5.1.2.56 make\_compiled\_model()**

```
template<typename Model , typename Optimizer , typename Loss >
auto ceras::make_compiled_model (
    Model const & m,
    Loss const & l,
    Optimizer const & o ) [inline]
```

**5.1.2.57 make\_trainable()**

```
template<Expression Ex>
void ceras::make_trainable (
    Ex & ex,
    bool t )
```

Setting an expression's trainable flag

**5.1.2.58 max() [1/2]**

```
template<Tensor Tsor>
Tsor ceras::max (
    Tsor const & ts,
    unsigned long axis,
    bool keepdims = false ) [noexcept]
```

**5.1.2.59 max() [2/2]**

```
template<Tensor Tsor>
auto ceras::max (
    Tsor const & tsor )
```

**5.1.2.60 MaxPooling2D()**

```
auto ceras::MaxPooling2D (
    unsigned long stride ) [inline], [noexcept]
```

Max pooling operation for 2D spatial data.

**5.1.2.61 mean() [1/3]**

```
template<Expression Ex>
constexpr auto ceras::mean (
    Ex const & ex ) [constexpr], [noexcept]
```

An alias name of `mean_reduce`.

**5.1.2.62 mean() [2/3]**

```
template<Tensor Tsor>
requires std::floating_point<typename Tsor::value_type> Tsor ceras::mean (
    Tsor const & ts,
    unsigned long axis,
    bool keepdims = false ) [noexcept]
```

**5.1.2.63 mean() [3/3]**

```
template<Tensor Tsor>
auto ceras::mean (
    Tsor const & tsor )
```

**5.1.2.64 mean\_absolute\_error()**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::mean_absolute_error (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

**5.1.2.65 mean\_reduce()**

```
template<Expression Ex>
constexpr auto ceras::mean_reduce (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes the mean of elements across all dimensions of an expression.

## Parameters

<i>ex</i>	Incoming expression.
-----------	----------------------

## Example code:

```
auto va = place_holder<tensor<float>>{};
auto vb = variable{ random<float>{ 3, 4 } };
auto diff = mean_reduce( va, vb );
```

## 5.1.2.66 mean\_squared\_error()

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::mean_squared_error (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

## 5.1.2.67 mean\_squared\_logarithmic\_error()

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::mean_squared_logarithmic_error (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

## 5.1.2.68 min() [1/2]

```
template<Tensor Tsor>
Tsor ceras::min (
    Tsor const & ts,
    unsigned long axis,
    bool keepdims = false ) [noexcept]
```

## 5.1.2.69 min() [2/2]

```
template<Tensor Tsor>
auto ceras::min (
    Tsor const & tsor )
```

**5.1.2.70 minus() [1/2]**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::minus (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

**5.1.2.71 minus() [2/2]**

```
template<Tensor Tsor>
Tsor ceras::minus (
    Tsor const & lhs,
    Tsor const & rhs ) [noexcept]
```

**5.1.2.72 mse()**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::mse (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

**5.1.2.73 Multiply()**

```
auto ceras::Multiply ( ) [inline], [noexcept]
```

Layer that elementwise multiplies two layers

Example usage:

```
auto input = Input(); // (16, )
auto x1 = Dense( 8, 16 )( input );
auto x2 = Dense( 8, 16 )( input );
auto x3 = Multiply()( x1, x2 ); // equivalent to 'elementwise_multiply(x1, x2)'
auto m = model{ input, x3 };
```

**5.1.2.74 multiply() [1/2]**

```
template<Tensor Tsor>
Tsor ceras::multiply (
    Tsor const & lhs,
    Tsor const & rhs ) [noexcept]
```

**5.1.2.75 multiply()** [2/2]

```
template<Tensor Tsor>
void ceras::multiply (
    Tsor const & lhs,
    Tsor const & rhs,
    Tsor & ans ) [noexcept]
```

**5.1.2.76 negative()**

```
template<Expression Ex>
constexpr auto ceras::negative (
    Ex const & ex ) [constexpr], [noexcept]
```

**5.1.2.77 negative\_relu()**

```
template<Expression Ex>
auto ceras::negative_relu (
    Ex const & ex ) [noexcept]
```

**5.1.2.78 norm()** [1/2]

```
template<Complex C>
auto ceras::norm (
    C const & c ) [noexcept]
```

Returns the squared magnitude of the complex expression.

**Parameters**

<i>c</i>	Complex expression.
----------	---------------------

```
auto r = variable{ ... };
auto i = variable{ ... };
auto c = complex{ r, i };
auto a = norm( c );
```

**5.1.2.79 norm()** [2/2]

```
template<Tensor Tsor>
auto ceras::norm (
    Tsor const & tsor )
```



**5.1.2.80 ones()**

```
template<typename T , typename A = default_allocator<T>>
constexpr tensor<T,A> ceras::ones (
    std::vector< unsigned long > const & shape ) [constexpr]
```

**5.1.2.81 ones\_like()**

```
template<Tensor Tsor>
constexpr Tsor ceras::ones_like (
    Tsor const & tsor ) [constexpr]
```

**5.1.2.82 operator"!="()**

```
template<Place_Holder Ph>
bool ceras::operator!= (
    Ph const & lhs,
    Ph const & rhs )
```

**5.1.2.83 operator\*() [1/7]**

```
template<Complex C, Expression E>
auto ceras::operator* (
    C const & c,
    E const & e ) [noexcept]
```

Multiplies a complex expression with an expression.

**5.1.2.84 operator\*() [2/7]**

```
template<Complex Cl, Complex Cr>
auto ceras::operator* (
    Cl const & cl,
    Cr const & cr ) [noexcept]
```

Multiplies two complex expressions. Optimization here:  $(a+ib)*(c+id) = (ac-bd) + i(ad+bc) = (ac-bd) + i((a+b)*(c+d)-ac-bd)$

```
auto c1 = complex{ ..., ... };
auto c2 = complex{ ..., ... };
auto c12 = c1 * c2;
```

**5.1.2.85 operator\*() [3/7]**

```
template<Complex C, Expression E>
auto ceras::operator* (
    E const & e,
    C const & c ) [noexcept]
```

Multiplies an expression with a compression expression.

**5.1.2.86 operator\*() [4/7]**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
auto ceras::operator* (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [noexcept]
```

**5.1.2.87 operator\*() [5/7]**

```
template<Tensor Tsor>
Tsor ceras::operator* (
    Tsor const & lhs,
    Tsor const & rhs ) [noexcept]
```

**5.1.2.88 operator\*() [6/7]**

```
template<Tensor Tsor>
Tsor ceras::operator* (
    Tsor const & lhs,
    typename Tsor::value_type const & rhs ) [noexcept]
```

**5.1.2.89 operator\*() [7/7]**

```
template<Tensor Tsor>
Tsor ceras::operator* (
    typename Tsor::value_type const & lhs,
    Tsor const & rhs ) [noexcept]
```

**5.1.2.90 operator+()** [1/9]

```
template<Complex C>
auto ceras::operator+ (
    C const & c ) [noexcept]
```

Returns the complex expression.

**5.1.2.91 operator+()** [2/9]

```
template<Complex C, Expression E>
auto ceras::operator+ (
    C const & c,
    E const & e ) [noexcept]
```

Sums up a complex expression and an expression.

**5.1.2.92 operator+()** [3/9]

```
template<Complex Cl, Complex Cr>
auto ceras::operator+ (
    Cl const & cl,
    Cr const & cr ) [noexcept]
```

Sums up two complex expressions.

**5.1.2.93 operator+()** [4/9]

```
template<Complex C, Expression E>
auto ceras::operator+ (
    E const & e,
    C const & c ) [noexcept]
```

Sums up a complex expression and an expression.

**5.1.2.94 operator+()** [5/9]

```
template<Expression Ex>
constexpr auto ceras::operator+ (
    Ex const & ex ) [constexpr], [noexcept]
```

**5.1.2.95 operator+()** [6/9]

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::operator+ (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

**5.1.2.96 operator+()** [7/9]

```
template<Tensor Tsor>
Tsor ceras::operator+ (
    Tsor const & lhs,
    Tsor const & rhs ) [noexcept]
```

**5.1.2.97 operator+()** [8/9]

```
template<Tensor Tsor>
Tsor ceras::operator+ (
    Tsor const & lhs,
    typename Tsor::value_type const & rhs ) [noexcept]
```

**5.1.2.98 operator+()** [9/9]

```
template<Tensor Tsor>
Tsor ceras::operator+ (
    typename Tsor::value_type const & lhs,
    Tsor const & rhs ) [noexcept]
```

**5.1.2.99 operator-()** [1/9]

```
template<Complex C>
auto ceras::operator- (
    C const & c ) [noexcept]
```

Negatives the complex expression.

**5.1.2.100 operator-() [2/9]**

```
template<Complex C, Expression E>
auto ceras::operator- (
    C const & c,
    E const & e ) [noexcept]
```

Subtracts an expression from a compression expression.

**5.1.2.101 operator-() [3/9]**

```
template<Complex Cl, Complex Cr>
auto ceras::operator- (
    Cl const & cl,
    Cr const & cr ) [noexcept]
```

Subtracts one complex expression from the other one.

**5.1.2.102 operator-() [4/9]**

```
template<Complex C, Expression E>
auto ceras::operator- (
    E const & e,
    C const & c ) [noexcept]
```

Subtracts a complex expression from an expression.

**5.1.2.103 operator-() [5/9]**

```
template<Expression Ex>
constexpr auto ceras::operator- (
    Ex const & ex ) [constexpr], [noexcept]
```

**5.1.2.104 operator-() [6/9]**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::operator- (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

**5.1.2.105 operator-() [7/9]**

```
template<Tensor Tsor>
Tsor ceras::operator- (
    Tsor const & lhs,
    Tsor const & rhs ) [noexcept]
```

**5.1.2.106 operator-() [8/9]**

```
template<Tensor Tsor>
Tsor ceras::operator- (
    Tsor const & lhs,
    typename Tsor::value_type const & rhs ) [noexcept]
```

**5.1.2.107 operator-() [9/9]**

```
template<Tensor Tsor>
Tsor ceras::operator- (
    typename Tsor::value_type const & lhs,
    Tsor const & rhs ) [noexcept]
```

**5.1.2.108 operator/()**

```
template<Tensor Tsor>
Tsor ceras::operator/ (
    Tsor const & lhs,
    typename Tsor::value_type const & rhs ) [noexcept]
```

**5.1.2.109 operator<()**

```
template<Place_Holder Ph>
bool ceras::operator< (
    Ph const & lhs,
    Ph const & rhs )
```

**5.1.2.110 operator<<()**

```
template<Tensor Tsor, typename CharT , typename Traits >
std::basic_ostream<CharT, Traits>& ceras::operator<< (
    std::basic_ostream< CharT, Traits > & os_,
    Tsor const & tsor )
```

**5.1.2.111 operator<=()**

```
template<Place_Holder Ph>
bool ceras::operator<= (
    Ph const & lhs,
    Ph const & rhs )
```

**5.1.2.112 operator==( ) [1/2]**

```
template<Place_Holder Ph>
bool ceras::operator==(
    Ph const & lhs,
    Ph const & rhs )
```

**5.1.2.113 operator==( ) [2/2]**

```
template<Variable Var>
bool ceras::operator==(
    Var const & lhs,
    Var const & rhs ) [noexcept]
```

**5.1.2.114 operator>()**

```
template<Place_Holder Ph>
bool ceras::operator> (
    Ph const & lhs,
    Ph const & rhs )
```

**5.1.2.115 operator>=()**

```
template<Place_Holder Ph>
bool ceras::operator>= (
    Ph const & lhs,
    Ph const & rhs )
```

**5.1.2.116 plus()**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::plus (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

### 5.1.2.117 polar()

```
template<Expression Em, Expression Ep>
auto ceras::polar (
    Em const & em,
    Ep const & ep ) [noexcept]
```

Returns with given magnitude and phase angle.

#### Parameters

<i>em</i>	Magnitude.
<i>ep</i>	Phase.

```
auto r = variable{ ... };
auto i = variable{ ... };
auto a = polar( r, i );
```

### 5.1.2.118 randn()

```
template<typename T , typename A = default_allocator<T>>
tensor<T,A> ceras::randn (
    std::vector< unsigned long > const & shape,
    T mean = T{0},
    T stddev = T{1} )
```

### 5.1.2.119 randn\_like()

```
template<Tensor Tsor>
Tsor ceras::randn_like (
    Tsor const & tsor,
    typename Tsor::value_type mean = 0,
    typename Tsor::value_type stddev = 1 )
```

### 5.1.2.120 random()

```
template<typename T , typename A = default_allocator<T>>
tensor<T,A> ceras::random (
    std::vector< unsigned long > const & shape,
    T min = T{0},
    T max = T{1} )
```



### 5.1.2.121 random\_like()

```
template<Tensor Tsor>
Tsor ceras::random_like (
    Tsor const & tsor,
    typename Tsor::value_type min = 0,
    typename Tsor::value_type max = 1 )
```

### 5.1.2.122 read\_tensor()

```
template<class _Tp , class _CharT , class _Traits , class _Alloc >
std::basic_istream<_CharT, _Traits>& ceras::read_tensor (
    std::basic_istream< _CharT, _Traits > & __is,
    tensor< _Tp, _Alloc > & __x )
```

### 5.1.2.123 real()

```
template<Expression Real_Ex, Expression Imag_Ex>
Real_Ex ceras::real (
    complex< Real_Ex, Imag_Ex > const & c ) [noexcept]
```

@bref Returns the real part of the complex expression.

#### Parameters

<i>c</i>	A complex expression.
----------	-----------------------

### 5.1.2.124 reduce()

```
template<Tensor Tsor, typename Function >
Tsor ceras::reduce (
    Tsor const & ts,
    unsigned long axis,
    typename Tsor::value_type const & init,
    Function const & func,
    bool keepdims = false ) [noexcept]
```

### 5.1.2.125 reduce\_mean() [1/2]

```
template<Expression Ex>
constexpr auto ceras::reduce_mean (
    Ex const & ex ) [constexpr], [noexcept]
```

An alias name of mean\_reduce.

**5.1.2.126 reduce\_mean()** [2/2]

```
template<Tensor Tsor>
Tsor ceras::reduce_mean (
    Tsor const & tsor )
```

**5.1.2.127 reduce\_sum()** [1/2]

```
template<Expression Ex>
constexpr auto ceras::reduce_sum (
    Ex const & ex ) [constexpr], [noexcept]
```

**5.1.2.128 reduce\_sum()** [2/2]

```
template<Tensor Tsor>
Tsor ceras::reduce_sum (
    Tsor const & tsor )
```

**5.1.2.129 relu()**

```
template<Expression Ex>
auto ceras::relu (
    Ex const & ex ) [noexcept]
```

**5.1.2.130 ReLU()**

```
template<Expression Ex>
auto ceras::ReLU (
    Ex const & ex ) [inline], [noexcept]
```

Rectified Linear Unit activation function.

**5.1.2.131 relu6()**

```
template<Expression Ex>
auto ceras::relu6 (
    Ex const & ex ) [noexcept]
```

### 5.1.2.132 repeat()

```
template<Tensor Tsor>
Tsor ceras::repeat (
    Tsor const & tsor,
    unsigned long n )
```

### 5.1.2.133 replace\_placeholder\_with\_expression()

```
template<Expression Ex, Place_Holder Ph, Expression Ey>
auto ceras::replace_placeholder_with_expression (
    Ex const & ex,
    Ph const & old_place_holder,
    Ey const & new_expression )
```

Replacing a [place\\_holder](#) with an expression.

#### Parameters

<i>ex</i>	Can be a unary operator, binary operator, variable, <a href="#">place_holder</a> , a constant or a value
<i>old_place_holder</i>	An place holder in <i>ex</i>
<i>new_expression</i>	An expression that will replace <i>old_place_holder</i> in <i>ex</i> .

#### Returns

A expression inheriting the topology of *ex*, but with *old\_place\_holder* replaced by *new\_expression*

### 5.1.2.134 repmat()

```
template<Tensor Tsor>
Tsor ceras::repmat (
    Tsor const & tsor,
    unsigned long row_rep,
    unsigned long col_rep )
```

### 5.1.2.135 Reshape()

```
auto ceras::Reshape (
    std::vector< unsigned long > const & new_shape,
    bool include_batch_flag = true ) [inline], [noexcept]
```

Reshapes inputs into the given shape.

#### 5.1.2.136 reshape()

```
template<Tensor Tsor>
Tsor ceras::reshape (
    Tsor const & ts,
    std::vector< unsigned long > const & new_shape )
```

#### 5.1.2.137 save\_tensor()

```
template<Tensor Tsor>
void ceras::save_tensor (
    std::string const & file_name,
    Tsor const & tsor )
```

#### 5.1.2.138 selu()

```
template<Expression Ex>
auto ceras::selu (
    Ex const & ex ) [inline], [noexcept]
```

#### 5.1.2.139 sigmoid()

```
template<Expression Ex>
auto ceras::sigmoid (
    Ex const & ex ) [inline], [noexcept]
```

#### 5.1.2.140 silu()

```
template<Expression Ex>
auto ceras::silu (
    Ex const & ex ) [noexcept]
```

An alias name of activation swish.

#### 5.1.2.141 Softmax()

```
auto ceras::Softmax ( ) [inline], [noexcept]
```

Softmax activation function.

**5.1.2.142 softmax()** [1/2]

```
template<Expression Ex>
constexpr auto ceras::softmax (
    Ex const & ex ) [constexpr], [noexcept]
```

**5.1.2.143 softmax()** [2/2]

```
template<Tensor Tsor>
Tsor ceras::softmax (
    Tsor const & tsor )
```

**5.1.2.144 softplus()**

```
template<Expression Ex>
auto ceras::softplus (
    Ex const & ex ) [inline], [noexcept]
```

**5.1.2.145 softsign()**

```
template<Expression Ex>
auto ceras::softsign (
    Ex const & ex ) [inline], [noexcept]
```

**5.1.2.146 square()**

```
template<Expression Ex>
constexpr auto ceras::square (
    Ex const & ex ) [constexpr], [noexcept]
```

Returns the square of the input

**Parameters**

<i>ex</i>	The input operator.
-----------	---------------------

**Returns**

An instance of a [unary\\_operator](#) that evaluate the squared value of the input operator.

Example code:

```
auto e = variable<tensor<float>>{ /*...*/ };
auto square = square(e);
```

#### 5.1.2.147 squared\_loss()

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::squared_loss (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

#### 5.1.2.148 squeeze()

```
template<Tensor Tsor>
Tsor ceras::squeeze (
    Tsor const & tsor )
```

#### 5.1.2.149 standard\_deviation()

```
template<Tensor Tsor>
requires std::floating_point<typename Tsor::value_type> Tsor ceras::standard_deviation (
    Tsor const & ts,
    unsigned long axis,
    bool keepdims = false ) [noexcept]
```

#### 5.1.2.150 std()

```
template<Tensor Tsor>
requires std::floating_point<typename Tsor::value_type> Tsor::value_type ceras::std (
    Tsor const & ts ) [noexcept]
```

#### 5.1.2.151 Subtract()

```
auto ceras::Subtract ( ) [inline], [noexcept]
```

Layer that subtracts two layers

Example usage:

```
auto input = Input(); // (16, )
auto x1 = Dense( 8, 16 )( input );
auto x2 = Dense( 8, 16 )( input );
auto x3 = Subtract()( x1, x2 ); // equivalent to 'x1 - x2'
auto m = model{ input, x3 };
```

**5.1.2.152 sum()** [1/2]

```
template<Tensor Tsor>
Tsor ceras::sum (
    Tsor const & ts,
    unsigned long axis,
    bool keepdims = false ) [noexcept]
```

**5.1.2.153 sum()** [2/2]

```
template<Tensor Tsor>
auto ceras::sum (
    Tsor const & tsor )
```

**5.1.2.154 sum\_reduce()**

```
template<Expression Ex>
constexpr auto ceras::sum_reduce (
    Ex const & ex ) [constexpr], [noexcept]
```

**5.1.2.155 swish()**

```
template<Expression Ex>
auto ceras::swish (
    Ex const & ex ) [noexcept]
```

Applies the swish activation function.

Reference: Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for Activation Functions." ArXiv:1710.05941 [Cs], October 16, 2017. <http://arxiv.org/abs/1710.05941>.

**Parameters**

<i>ex</i>	Input expression.
-----------	-------------------

**5.1.2.156 truncated\_normal()**

```
template<typename T , typename A = default_allocator<T>>
tensor<T,A> ceras::truncated_normal (
    std::vector< unsigned long > const & shape,
```

```

T mean = T{0},
T stddev = T{1},
T lower = T{0},
T upper = T{1} )

```

#### 5.1.2.157 update\_cuda\_gemm\_threshold()

```
void ceras::update_cuda_gemm_threshold ( ) [inline]
```

#### 5.1.2.158 UpSampling2D()

```
auto ceras::UpSampling2D (
    unsigned long stride ) [inline], [noexcept]
```

Upsampling layer for 2D inputs.

#### 5.1.2.159 var()

```
template<Tensor Tsor>
requires std::floating_point<typename Tsor::value_type> Tsor::value_type ceras::var (
    Tsor const & ts ) [noexcept]
```

#### 5.1.2.160 variance()

```
template<Tensor Tsor>
requires std::floating_point<typename Tsor::value_type> Tsor ceras::variance (
    Tsor const & ts,
    unsigned long axis,
    bool keepdims = false ) [noexcept]
```

#### 5.1.2.161 write\_tensor()

```
template<class _Tp , class _CharT , class _Traits , class _Alloc >
std::basic_ostream<_CharT, _Traits>& ceras::write_tensor (
    std::basic_ostream< _CharT, _Traits > & __os,
    tensor< _Tp, _Alloc > const & __x )
```



### 5.1.2.162 zeros()

```
template<typename T , typename A = default_allocator<T>>
constexpr tensor<T,A> ceras::zeros (
    std::vector< unsigned long > const & shape ) [constexpr]
```

### 5.1.2.163 zeros\_like()

```
template<Tensor Tsor>
constexpr Tsor ceras::zeros_like (
    Tsor const & tsor ) [constexpr]
```

## 5.1.3 Variable Documentation

### 5.1.3.1 Adadelta

```
auto ceras::Adadelta [inline]
```

#### Initial value:

```
= [] ( auto ... args )
{
    return [=]<Expression Ex>( Ex& loss )
    {
        return adadelta{loss, args...};
    };
}
```

### 5.1.3.2 Adagrad

```
auto ceras::Adagrad [inline]
```

#### Initial value:

```
= [] ( auto ... args )
{
    return [=]<Expression Ex>( Ex& loss )
    {
        return adagrad{loss, args...};
    };
}
```

### 5.1.3.3 Adam

```
auto ceras::Adam [inline]
```

#### Initial value:

```
= [] ( auto ... args )
{
    return [=]<Expression Ex>( Ex& loss )
    {
        return adam{loss, args...};
    };
}
```

#### 5.1.3.4 Binary\_Operator

```
template<typename T >
concept ceras::Binary_Operator = is_binary_operator_v<T>
```

A type that represents a binary operator.

```
@concept Binary_Operator<>
```

#### 5.1.3.5 BinaryCrossentropy

```
auto ceras::BinaryCrossentropy [inline]
```

**Initial value:**

```
= [] ()
{
    return []<Expression Ex >( Ex const& output )
    {
        return [=]<Place_Holder Ph>( Ph const& ground_truth )
        {
            return binary_cross_entropy_loss( ground_truth, output );
        };
    };
}
```

#### 5.1.3.6 BinaryCrossEntropy

```
auto ceras::BinaryCrossEntropy [inline]
```

**Initial value:**

```
= [] ()
{
    return BinaryCrossentropy();
}
```

#### 5.1.3.7 CategoricalCrossentropy

```
auto ceras::CategoricalCrossentropy [inline]
```

**Initial value:**

```
= [] ()
{
    return []<Expression Ex >( Ex const& output )
    {
        return [=]<Place_Holder Ph>( Ph const& ground_truth )
        {
            return cross_entropy_loss( ground_truth, output );
        };
    };
}
```

### 5.1.3.8 CategoricalCrossEntropy

```
auto ceras::CategoricalCrossEntropy [inline]
```

**Initial value:**

```
= [] ()
{
    return CategoricalCrossentropy();
}
```

### 5.1.3.9 Complex

```
template<typename T >
concept ceras::Complex = is_complex_v<T>
```

A type that represents a complex expression.

@concept Complex

### 5.1.3.10 Constant

```
template<typename T >
concept ceras::Constant = is_constant_v<T>
```

### 5.1.3.11 Expression

```
template<typename T >
concept ceras::Expression = Operator<T> || Variable<T> || Place_Holder<T> || Constant<T> ||
Value<T>
```

A type that represents a unary operator, a binary operator, a variable, a [place\\_holder](#), a constant or a value.

@concept Expression<>

### 5.1.3.12 Hinge

```
auto ceras::Hinge [inline]
```

**Initial value:**

```
= [] ()
{
    return []<Expression Ex >( Ex const& output )
    {
        return []<Place_Holder Ph>( Ph const& ground_truth )
        {
            return hinge_loss( ground_truth, output );
        };
    };
}
```

#### 5.1.3.13 is\_binary\_operator\_v

```
template<class T >
constexpr bool ceras::is_binary_operator_v = is_binary_operator<T>::value [inline], [constexpr]
```

If T is an instance of a [binary\\_operator](#), the constant value equals to `true`. Otherwise this value is `false`.

#### 5.1.3.14 is\_complex\_v

```
template<typename T >
constexpr bool ceras::is_complex_v = is_complex<T>::value [constexpr]
```

#### 5.1.3.15 is\_constant\_v

```
template<class T >
constexpr bool ceras::is_constant_v = is_constant<T>::value [inline], [constexpr]
```

#### 5.1.3.16 is\_place\_holder\_v

```
template<class T >
constexpr bool ceras::is_place_holder_v = is_place_holder<T>::value [inline], [constexpr]
```

#### 5.1.3.17 is\_tensor\_v

```
template<class T >
constexpr bool ceras::is_tensor_v = is_tensor<T>::value [inline], [constexpr]
```

#### 5.1.3.18 is\_unary\_operator\_v

```
template<class T >
constexpr bool ceras::is_unary_operator_v = is_unary_operator<T>::value [inline], [constexpr]
```

If T is an instance of a [unary\\_operator](#), the constant value equals to `true`. Otherwise this value is `false`.

#### 5.1.3.19 is\_value\_v

```
template<class T >
constexpr bool ceras::is_value_v = is_value<T>::value [inline], [constexpr]
```

### 5.1.3.20 is\_variable\_v

```
template<class T >
constexpr bool ceras::is_variable_v = is_variable<T>::value [inline], [constexpr]
```

### 5.1.3.21 MAE

```
auto ceras::MAE [inline]
```

#### Initial value:

```
= [] ()
{
    return MeanAbsoluteError();
}
```

An alias name of function [MeanAbsoluteError](#).

### 5.1.3.22 make\_binary\_operator

```
constexpr auto ceras::make_binary_operator [static], [constexpr]
```

#### Initial value:

```
= [] ( auto const& binary_forward_action, auto const& binary_backward_action, std::string const&
    name="Anonymous Binary Operator" ) noexcept
{
    return [&binary_forward_action, &binary_backward_action, &name]( auto const& lhs_op, auto const&
    rhs_op ) noexcept
    {
        auto ans = binary_operator{ lhs_op, rhs_op, binary_forward_action, binary_backward_action };
        ans.name_ = name;
        return ans;
    };
}
```

### 5.1.3.23 make\_unary\_operator

```
constexpr auto ceras::make_unary_operator [static], [constexpr]
```

#### Initial value:

```
= [] ( auto const& unary_forward_action, auto const& unary_backward_action, std::string const&
    name="Anonymous Unary Operator" ) noexcept
{
    return [&unary_forward_action, &unary_backward_action, &name]( auto const& op ) noexcept
    {
        auto ans = unary_operator{ op, unary_forward_action, unary_backward_action };
        ans.name_ = name;
        return ans;
    };
}
```

### 5.1.3.24 MeanAbsoluteError

```
auto ceras::MeanAbsoluteError [inline]
```

**Initial value:**

```
= [] ()
{
    return []<Expression Ex >( Ex const& output )
    {
        return [=]<Place_Holder Ph>( Ph const& ground_truth )
        {
            return mean_absolute_error( ground_truth, output );
        };
    };
}
```

Computes the mean of absolute errors between labels and predictions.

```
auto input = place_holder<tensor<float>>{};
auto v = variable<tensor<float>>{ ones<float>({12, 34}) };
auto output = input * v;
auto m = model{ input, output };
auto cm = m.compile( MeanAbsoluteError(), Adam(128/*batch size*/, 0.01f/*learning rate*/) );
```

see also [mean\\_absolute\\_error](#)

### 5.1.3.25 MeanSquaredError

```
auto ceras::MeanSquaredError [inline]
```

**Initial value:**

```
= [] ()
{
    return []<Expression Ex >( Ex const& output )
    {
        return [=]<Place_Holder Ph>( Ph const& ground_truth )
        {
            return mean_squared_error( ground_truth, output );
        };
    };
}
```

Computes the mean of squares of errors between labels and predictions.

```
auto input = place_holder<tensor<float>>{};
auto v = variable<tensor<float>>{ ones<float>({12, 34}) };
auto output = input * v;
auto m = model{ input, output };
auto cm = m.compile( MeanSquaredError(), Adam(128/*batch size*/, 0.01f/*learning rate*/) );
```

see also [mean\\_squared\\_error](#)

### 5.1.3.26 MSE

```
auto ceras::MSE [inline]
```

**Initial value:**

```
= [] ()
{
    return MeanSquaredError();
}
```

An alias name of function [MeanSquaredError](#).

### 5.1.3.27 Operator

```
template<typename T >
concept ceras::Operator = Unary_Operator<T> || Binary_Operator<T>
```

A type that represents an unary or a binary operator.

```
@concept Operator<>
```

### 5.1.3.28 Place\_Holder

```
template<typename T >
concept ceras::Place_Holder = is_place_holder_v<T>
```

### 5.1.3.29 random\_generator

```
std::mt19937 ceras::random_generator {random_seed} [static]
```

### 5.1.3.30 random\_seed

```
unsigned long ceras::random_seed = std::chrono::system_clock::now().time_since_epoch().count()
[static]
```

### 5.1.3.31 RMSprop

```
auto ceras::RMSprop [inline]
```

**Initial value:**

```
= [] ( auto ... args )
{
    return [=]<Expression Ex>( Ex& loss )
    {
        return rmsprop{loss, args...};
    };
}
```

### 5.1.3.32 SGD

```
auto ceras::SGD [inline]
```

**Initial value:**

```
= [] ( auto ... args )
{
    return [=]<Expression Ex>( Ex& loss )
    {
        return sgd{loss, args...};
    };
}
```

### 5.1.3.33 Tensor

```
template<typename T >  
concept ceras::Tensor = is\_tensor\_v<T>
```

### 5.1.3.34 Unary\_Operator

```
template<typename T >  
concept ceras::Unary_Operator = is\_unary\_operator\_v<T>
```

A type that represents an unary operator.

@concept Unary\_Operator<>

### 5.1.3.35 Value

```
template<typename T >  
concept ceras::Value = is\_value\_v<T>
```

### 5.1.3.36 Variable

```
template<typename T >  
concept ceras::Variable = is\_variable\_v<T>
```

## 5.2 ceras::ceras\_private Namespace Reference

### Classes

- struct [session](#)

## 5.3 ceras::dataset Namespace Reference

### Namespaces

- [fashion\\_mnist](#)
- [mnist](#)

## 5.4 ceras::dataset::fashion\_mnist Namespace Reference

### Functions

- auto [load\\_data](#) (std::string const &path=std::string{"/dataset/fashion\_mnist"})



## 5.4.1 Function Documentation

### 5.4.1.1 load\_data()

```
auto ceras::dataset::fashion_mnist::load_data (
    std::string const & path = std::string{"./dataset/fashion_mnist"} ) [inline]
```

Loads the fashion-MNIST dataset.

#### Parameters

<i>path</i>	Path where to cache the dataset locally. Default to <code>./dataset/fashion_mnist</code> , should be updated if running the program somewhere else.
-------------	---

#### Returns

A tuple of 4 tensors: `x_train`, `y_train`, `x_test`, `y_test`. `x_train`, `x_test`: uint8 arrays of grayscale image data with shapes (num\_samples, 28, 28). `y_train`, `y_test`: uint8 tensor of digit labels (integers in range 0-9) with shapes (num\_samples, 10). Note: for digit 0, the corresponding array is `[[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]]`.

Label Description 0 T-shirt/top 1 Trouser 2 Pullover 3 Dress 4 Coat 5 Sandal 6 Shirt 7 Sneaker 8 Bag 9 Ankle boot

#### Example usage:

```
auto const& [x_train, y_train, x_test, y_test] =
    ceras::dataset::mnist::load_data("/home/feng/dataset/fashion_mnist");
```

The copyright for Fashion-MNIST is held by Zalando SE. Fashion-MNIST is licensed under the MIT license.

## 5.5 ceras::dataset::mnist Namespace Reference

### Functions

- auto [load\\_data](#) (std::string const &path=std::string{"./dataset/mnist"})

### 5.5.1 Function Documentation

#### 5.5.1.1 load\_data()

```
auto ceras::dataset::mnist::load_data (
    std::string const & path = std::string{"./dataset/mnist"} ) [inline]
```

Loads the MNIST dataset.

### Parameters

<i>path</i>	Path where to cache the dataset locally. Default to <code>"/dataset/mnist"</code> , should be updated if running the program somewhere else.
-------------	--

### Returns

A tuple of 4 tensors: `x_train`, `y_train`, `x_test`, `y_test`. `x_train`, `x_test`: uint8 arrays of grayscale image data with shapes `(num_samples, 28, 28)`. `y_train`, `y_test`: uint8 tensor of digit labels (integers in range 0-9) with shapes `(num_samples, 10)`. Note: for digit 0, the corresponding array is `[[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]]`.

### Example usage:

```
auto const& [x_train, y_train, x_test, y_test] =  
    ceras::dataset::mnist::load_data("/home/feng/dataset/mnist");
```

Yann LeCun and Corinna Cortes hold the copyright of MNIST dataset, which is available under the terms of the Creative Commons Attribution-Share Alike 3.0 license.

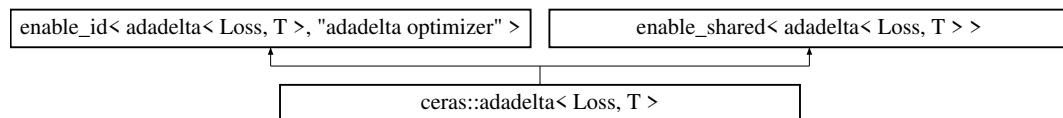
## Chapter 6

# Class Documentation

### 6.1 `ceras::adadelta< Loss, T >` Struct Template Reference

```
#include <optimizer.hpp>
```

Inheritance diagram for `ceras::adadelta< Loss, T >`:



#### Public Types

- typedef `tensor< T >` `tensor_type`

#### Public Member Functions

- `adadelta` (`Loss &loss`, `std::size_t batch_size`, `T rho=0.9`) `noexcept`
- void `forward` ()

#### Public Attributes

- `Loss &` `loss_`
- `T` `rho_`
- `T` `learning_rate_`
- unsigned long `iterations_`

#### 6.1.1 Member Typedef Documentation

### 6.1.1.1 tensor\_type

```
template<typename Loss , typename T >
typedef tensor< T > ceras::adadelta< Loss, T >::tensor_type
```

## 6.1.2 Constructor & Destructor Documentation

### 6.1.2.1 adadelta()

```
template<typename Loss , typename T >
ceras::adadelta< Loss, T >::adadelta (
    Loss & loss,
    std::size_t batch_size,
    T rho = 0.9 ) [inline], [noexcept]
```

## 6.1.3 Member Function Documentation

### 6.1.3.1 forward()

```
template<typename Loss , typename T >
void ceras::adadelta< Loss, T >::forward ( ) [inline]
```

## 6.1.4 Member Data Documentation

### 6.1.4.1 iterations\_

```
template<typename Loss , typename T >
unsigned long ceras::adadelta< Loss, T >::iterations_
```

### 6.1.4.2 learning\_rate\_

```
template<typename Loss , typename T >
T ceras::adadelta< Loss, T >::learning_rate_
```

### 6.1.4.3 loss\_

```
template<typename Loss , typename T >
Loss& ceras::adadelat< Loss, T >::loss_
```

### 6.1.4.4 rho\_

```
template<typename Loss , typename T >
T ceras::adadelat< Loss, T >::rho_
```

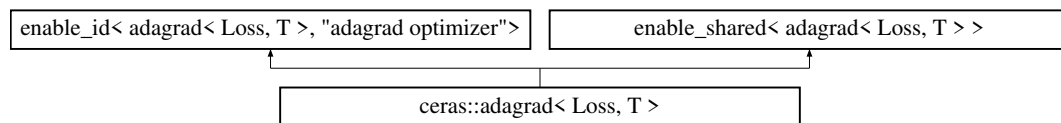
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/optimizer.hpp>

## 6.2 ceras::adagrad< Loss, T > Struct Template Reference

```
#include <optimizer.hpp>
```

Inheritance diagram for ceras::adagrad< Loss, T >:



### Public Types

- typedef [tensor](#)< T > [tensor\\_type](#)

### Public Member Functions

- [adagrad](#) (Loss &loss, std::size\_t batch\_size, T learning\_rate=1.0e-1, T decay=0.0) noexcept
- void [forward](#) ()

### Public Attributes

- Loss & [loss\\_](#)
- T [learning\\_rate\\_](#)
- T [decay\\_](#)
- unsigned long [iterations\\_](#)

### 6.2.1 Member Typedef Documentation

### 6.2.1.1 tensor\_type

```
template<typename Loss , typename T >
typedef tensor< T > ceras::adagrad< Loss, T >::tensor\_type
```

## 6.2.2 Constructor & Destructor Documentation

### 6.2.2.1 adagrad()

```
template<typename Loss , typename T >
ceras::adagrad< Loss, T >::adagrad (
    Loss & loss,
    std::size_t batch_size,
    T learning_rate = 1.0e-1,
    T decay = 0.0 ) [inline], [noexcept]
```

## 6.2.3 Member Function Documentation

### 6.2.3.1 forward()

```
template<typename Loss , typename T >
void ceras::adagrad< Loss, T >::forward ( ) [inline]
```

## 6.2.4 Member Data Documentation

### 6.2.4.1 decay\_

```
template<typename Loss , typename T >
T ceras::adagrad< Loss, T >::decay_
```

### 6.2.4.2 iterations\_

```
template<typename Loss , typename T >
unsigned long ceras::adagrad< Loss, T >::iterations_
```

#### 6.2.4.3 learning\_rate\_

```
template<typename Loss , typename T >
T ceras::adagrad< Loss, T >::learning_rate_
```

#### 6.2.4.4 loss\_

```
template<typename Loss , typename T >
Loss& ceras::adagrad< Loss, T >::loss_
```

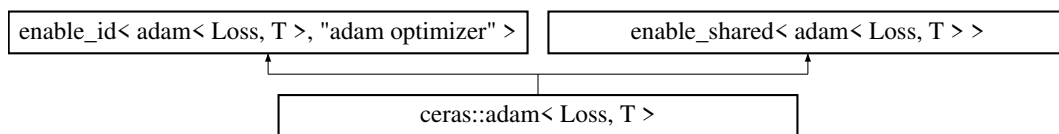
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/optimizer.hpp>

## 6.3 ceras::adam< Loss, T > Struct Template Reference

```
#include <optimizer.hpp>
```

Inheritance diagram for ceras::adam< Loss, T >:



### Public Types

- typedef [tensor](#)< T > [tensor\\_type](#)

### Public Member Functions

- [adam](#) (Loss &loss, std::size\_t batch\_size, T learning\_rate=1.0e-1, T beta\_1=0.9, T beta\_2=0.999, bool amsgrad=false) noexcept
- void [forward](#) ()

### Public Attributes

- Loss & [loss\\_](#)
- T [learning\\_rate\\_](#)
- T [beta\\_1\\_](#)
- T [beta\\_2\\_](#)
- bool [amsgrad\\_](#)
- unsigned long [iterations\\_](#)

## 6.3.1 Member Typedef Documentation

### 6.3.1.1 tensor\_type

```
template<typename Loss , typename T >
typedef tensor< T > ceras::adam< Loss, T >::tensor\_type
```

## 6.3.2 Constructor & Destructor Documentation

### 6.3.2.1 adam()

```
template<typename Loss , typename T >
ceras::adam< Loss, T >::adam (
    Loss & loss,
    std::size_t batch_size,
    T learning_rate = 1.0e-1,
    T beta_1 = 0.9,
    T beta_2 = 0.999,
    bool amsgrad = false ) [inline], [noexcept]
```

## 6.3.3 Member Function Documentation

### 6.3.3.1 forward()

```
template<typename Loss , typename T >
void ceras::adam< Loss, T >::forward ( ) [inline]
```

## 6.3.4 Member Data Documentation

### 6.3.4.1 amsgrad\_

```
template<typename Loss , typename T >
bool ceras::adam< Loss, T >::amsgrad\_
```



#### 6.3.4.2 beta\_1\_

```
template<typename Loss , typename T >
T ceras::adam< Loss, T >::beta_1_
```

#### 6.3.4.3 beta\_2\_

```
template<typename Loss , typename T >
T ceras::adam< Loss, T >::beta_2_
```

#### 6.3.4.4 iterations\_

```
template<typename Loss , typename T >
unsigned long ceras::adam< Loss, T >::iterations_
```

#### 6.3.4.5 learning\_rate\_

```
template<typename Loss , typename T >
T ceras::adam< Loss, T >::learning_rate_
```

#### 6.3.4.6 loss\_

```
template<typename Loss , typename T >
Loss& ceras::adam< Loss, T >::loss_
```

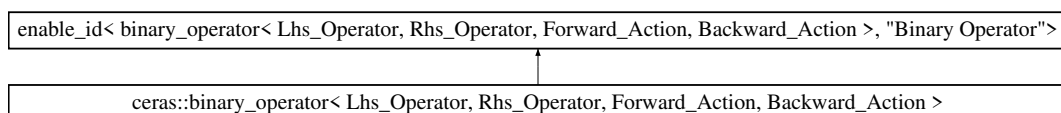
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/optimizer.hpp>

## 6.4 ceras::binary\_operator< Lhs\_Operator, Rh Operator, Forward\_Action, Backward\_Action > Struct Template Reference

```
#include <operation.hpp>
```

Inheritance diagram for ceras::binary\_operator< Lhs\_Operator, Rh Operator, Forward\_Action, Backward\_Action >:



## Public Types

- typedef [tensor\\_deduction](#)< Lhs\_Operator, Rhs\_Operator >::[tensor\\_type](#) [tensor\\_type](#)

## Public Member Functions

- [binary\\_operator](#) (Lhs\_Operator const &lhs\_op, Rhs\_Operator const &rhs\_op, Forward\_Action const &forward\_action, Backward\_Action const &backward\_action) noexcept
- auto [forward](#) ()
- void [backward](#) ([tensor\\_type](#) const &grad)

## Public Attributes

- Lhs\_Operator [lhs\\_op\\_](#)
- Rhs\_Operator [rhs\\_op\\_](#)
- Forward\_Action [forward\\_action\\_](#)
- Backward\_Action [backward\\_action\\_](#)
- [tensor\\_type](#) [lhs\\_input\\_data\\_](#)
- [tensor\\_type](#) [rhs\\_input\\_data\\_](#)
- [tensor\\_type](#) [output\\_data\\_](#)

## 6.4.1 Member Typedef Documentation

### 6.4.1.1 [tensor\\_type](#)

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
typedef tensor\_deduction<Lhs_Operator, Rhs_Operator>::tensor\_type ceras::binary\_operator<
Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >::tensor\_type
```

## 6.4.2 Constructor & Destructor Documentation

### 6.4.2.1 [binary\\_operator](#)()

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
ceras::binary\_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >::binary\_operator
(
    Lhs_Operator const & lhs_op,
    Rhs_Operator const & rhs_op,
    Forward_Action const & forward_action,
    Backward_Action const & backward_action ) [inline], [noexcept]
```

### 6.4.3 Member Function Documentation

#### 6.4.3.1 backward()

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
void ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >↵
::backward (
    tensor_type const & grad ) [inline]
```

#### 6.4.3.2 forward()

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
auto ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >↵
::forward ( ) [inline]
```

### 6.4.4 Member Data Documentation

#### 6.4.4.1 backward\_action\_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
Backward_Action ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward↵
_Action >::backward_action_
```

#### 6.4.4.2 forward\_action\_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
Forward_Action ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward↵
_Action >::forward_action_
```

#### 6.4.4.3 lhs\_input\_data\_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
tensor_type ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward↵
_Action >::lhs_input_data_
```

#### 6.4.4.4 lhs\_op\_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
Lhs_Operator ceras::binary\_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_↵
Action >::lhs_op_
```

#### 6.4.4.5 output\_data\_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
tensor\_type ceras::binary\_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_↵
Action >::output_data_
```

#### 6.4.4.6 rhs\_input\_data\_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
tensor\_type ceras::binary\_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_↵
Action >::rhs_input_data_
```

#### 6.4.4.7 rhs\_op\_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
Rhs_Operator ceras::binary\_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_↵
Action >::rhs_op_
```

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

- [/home/feng/workspace/github.repo/ceras/include/operation.hpp](#)

## 6.5 [ceras::compiled\\_model](#)< Model, Optimizer, Loss > Struct Template Reference

```
#include <model.hpp>
```

### Public Types

- typedef Model::input\_layer\_type [io\\_layer\\_type](#)

## Public Member Functions

- `compiled_model` (Model const &m, `io_layer_type` const &input\_place\_holder, `io_layer_type` const &ground\_truth\_place\_holder, Loss const &loss, Optimizer const &optimizer)
- `template<Tensor Tsr>`  
auto `evaluate` (Tsr const &inputs, Tsr const &outputs, unsigned long batch\_size=32)
- `template<Tensor Tsr>`  
auto `fit` (Tsr const &inputs, Tsr const &outputs, unsigned long batch\_size, unsigned long epoch=1, int verbose=0, double validation\_split=0.0)
- `template<Tensor Tsr>`  
auto `train_on_batch` (Tsr const &input, Tsr const &output)
- `template<Tensor Tsr>`  
auto `predict` (Tsr const &input\_tensor)
- `template<Expression Exp>`  
auto `operator()` (Exp const &ex) const noexcept
- void `trainable` (bool t)

## Public Attributes

- `decltype(std::declval< Optimizer >())(std::declval< Loss & >())` typedef `optimizer_type`
- Model `model_`
- `io_layer_type` `input_place_holder_`
- `io_layer_type` `ground_truth_place_holder_`
- Loss `loss_`
- Optimizer `optimizer_`
- `optimizer_type` `compiled_optimizer_`

## 6.5.1 Member Typedef Documentation

### 6.5.1.1 io\_layer\_type

```
template<typename Model , typename Optimizer , typename Loss >
typedef Model::input_layer_type ceras::compiled_model< Model, Optimizer, Loss >::io_layer_type
```

## 6.5.2 Constructor & Destructor Documentation

### 6.5.2.1 compiled\_model()

```
template<typename Model , typename Optimizer , typename Loss >
ceras::compiled_model< Model, Optimizer, Loss >::compiled_model (
    Model const & m,
    io_layer_type const & input_place_holder,
    io_layer_type const & ground_truth_place_holder,
    Loss const & loss,
    Optimizer const & optimizer ) [inline]
```

## 6.5.3 Member Function Documentation

### 6.5.3.1 evaluate()

```
template<typename Model , typename Optimizer , typename Loss >
template<Tensor Tsor>
auto ceras::compiled_model< Model, Optimizer, Loss >::evaluate (
    Tsor const & inputs,
    Tsor const & outputs,
    unsigned long batch_size = 32 ) [inline]
```

Calculate the loss for the model in test model.

#### Parameters

<i>inputs</i>	Input data. A tensor of shape (samples, input_shape).
<i>outputs</i>	Output data. A tensor of shape (samples, output_shape).
<i>batch_size</i>	Number of samples per batch of computation. Default to 32.

#### Returns

Test loss. A scalar.

### 6.5.3.2 fit()

```
template<typename Model , typename Optimizer , typename Loss >
template<Tensor Tsor>
auto ceras::compiled_model< Model, Optimizer, Loss >::fit (
    Tsor const & inputs,
    Tsor const & outputs,
    unsigned long batch_size,
    unsigned long epoch = 1,
    int verbose = 0,
    double validation_split = 0.0 ) [inline]
```

Train the model on the selected dataset for a fixed numbers of epoches.

#### Parameters

<i>inputs</i>	Input data. A tensor of shape (samples, input_shape).
<i>outputs</i>	Input data. A tensor of shape (samples, output_shape).
<i>batch_size</i>	Number of samples per gradient update. Should agree with the batch size in the optimizer.
<i>epoch</i>	Number of epoches to train the dataset.
<i>verbose</i>	Verbosity mode. 0 for silent. 1 for one line per epoch.
<i>validation_split</i>	Fraction of the training data that will be used for validation. A floating number in range [0, 1].

### Returns

A tuple of two vectors. The first vector gives the historical errors on the training data. The second vector gives the historical errors on the validation data.

#### Example:

```
model m{ ... };
auto cm = m.compile( ... );
tensor<float> inputs, outputs;
//...
unsigned long batch_size = 32;
unsigned long epoch = 10;
int verbose = 1;
double validation_split = 0.2;
auto errors = cm.fit( inputs, outputs, batch_size, epoch, verbose, validation_split );
```

#### 6.5.3.3 operator()

```
template<typename Model , typename Optimizer , typename Loss >
template<Expression Exp>
auto ceras::compiled_model< Model, Optimizer, Loss >::operator() (
    Exp const & ex ) const [inline], [noexcept]
```

#### 6.5.3.4 predict()

```
template<typename Model , typename Optimizer , typename Loss >
template<Tensor Tsor>
auto ceras::compiled_model< Model, Optimizer, Loss >::predict (
    Tsor const & input_tensor ) [inline]
```

#### 6.5.3.5 train\_on\_batch()

```
template<typename Model , typename Optimizer , typename Loss >
template<Tensor Tsor>
auto ceras::compiled_model< Model, Optimizer, Loss >::train_on_batch (
    Tsor const & input,
    Tsor const & output ) [inline]
```

Running a single updated on a single batch of data.

### Parameters

<i>input</i>	The input data to train the model. A tensor of shape (batch_size, input_shape).
<i>output</i>	The output data to train the model. A tensor of shape (batch_size, output_shape).

### Returns

Training loss. A scalar.

Example code:

```
auto m = model{ ... };
auto cm = m.compile( ... );
for ( auto idx : range( 1024 ) )
{
    auto x = ...; // get batch input
    auto y = ...; // get batch output
    cm.train_on_batch( x, y );
}
```

### 6.5.3.6 trainable()

```
template<typename Model , typename Optimizer , typename Loss >
void ceras::compiled_model< Model, Optimizer, Loss >::trainable (
    bool t ) [inline]
```

## 6.5.4 Member Data Documentation

### 6.5.4.1 compiled\_optimizer\_

```
template<typename Model , typename Optimizer , typename Loss >
optimizer_type ceras::compiled_model< Model, Optimizer, Loss >::compiled_optimizer_
```

### 6.5.4.2 ground\_truth\_place\_holder\_

```
template<typename Model , typename Optimizer , typename Loss >
io_layer_type ceras::compiled_model< Model, Optimizer, Loss >::ground_truth_place_holder_
```

### 6.5.4.3 input\_place\_holder\_

```
template<typename Model , typename Optimizer , typename Loss >
io_layer_type ceras::compiled_model< Model, Optimizer, Loss >::input_place_holder_
```

### 6.5.4.4 loss\_

```
template<typename Model , typename Optimizer , typename Loss >
Loss ceras::compiled_model< Model, Optimizer, Loss >::loss_
```



#### 6.5.4.5 model\_

```
template<typename Model , typename Optimizer , typename Loss >
Model ceras::compiled_model< Model, Optimizer, Loss >::model_
```

#### 6.5.4.6 optimizer\_

```
template<typename Model , typename Optimizer , typename Loss >
Optimizer ceras::compiled_model< Model, Optimizer, Loss >::optimizer_
```

#### 6.5.4.7 optimizer\_type

```
template<typename Model , typename Optimizer , typename Loss >
decltype(std::declval<Optimizer>() (std::declval<Loss&>())) typedef ceras::compiled_model<
Model, Optimizer, Loss >::optimizer_type
```

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

- </home/feng/workspace/github.repo/ceras/include/model.hpp>

## 6.6 ceras::complex< Real\_Ex, Imag\_Ex > Struct Template Reference

```
#include <complex_operator.hpp>
```

### Public Attributes

- Real\_Ex [real\\_](#)
- Imag\_Ex [imag\\_](#)

### 6.6.1 Member Data Documentation

#### 6.6.1.1 imag\_

```
template<Expression Real_Ex, Expression Imag_Ex>
Imag_Ex ceras::complex< Real_Ex, Imag_Ex >::imag_
```

### 6.6.1.2 real\_

```
template<Expression Real_Ex, Expression Imag_Ex>
Real_Ex ceras::complex< Real_Ex, Imag_Ex >::real_
```

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

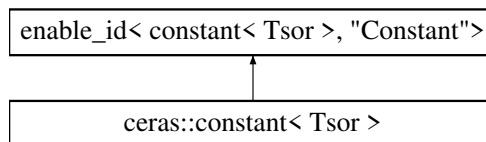
- [/home/feng/workspace/github.repo/ceras/include/complex\\_operator.hpp](/home/feng/workspace/github.repo/ceras/include/complex_operator.hpp)

## 6.7 ceras::constant< Tzor > Struct Template Reference

Creates a constant expression from a tensor-like object.

```
#include <constant.hpp>
```

Inheritance diagram for ceras::constant< Tzor >:



### Public Member Functions

- [constant](#) (Tzor const &data)
- void [backward](#) (auto) const
- Tzor [forward](#) () const
- auto [shape](#) () const

### Public Attributes

- Tzor [data\\_](#)

### 6.7.1 Detailed Description

```
template<Tensor Tzor>
struct ceras::constant< Tzor >
```

Creates a constant expression from a tensor-like object.

```
auto c = constant{ zeros<float>( {3, 3, 3} ) };
```

### 6.7.2 Constructor & Destructor Documentation

### 6.7.2.1 constant()

```
template<Tensor Tzor>
ceras::constant< Tzor >::constant (
    Tzor const & data ) [inline]
```

## 6.7.3 Member Function Documentation

### 6.7.3.1 backward()

```
template<Tensor Tzor>
void ceras::constant< Tzor >::backward (
    auto ) const [inline]
```

### 6.7.3.2 forward()

```
template<Tensor Tzor>
Tzor ceras::constant< Tzor >::forward ( ) const [inline]
```

### 6.7.3.3 shape()

```
template<Tensor Tzor>
auto ceras::constant< Tzor >::shape ( ) const [inline]
```

## 6.7.4 Member Data Documentation

### 6.7.4.1 data\_

```
template<Tensor Tzor>
Tzor ceras::constant< Tzor >::data_
```

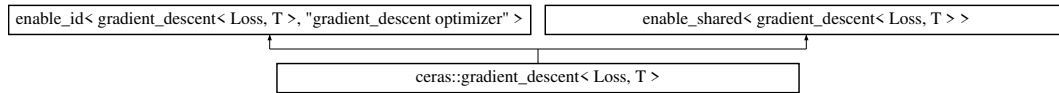
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/constant.hpp>

## 6.8 ceras::gradient\_descent< Loss, T > Struct Template Reference

```
#include <optimizer.hpp>
```

Inheritance diagram for ceras::gradient\_descent< Loss, T >:



### Public Types

- typedef [tensor](#)< T > [tensor\\_type](#)

### Public Member Functions

- [gradient\\_descent](#) (Loss &loss, std::size\_t batch\_size, T learning\_rate=1.0e-3, T momentum=0.0) noexcept
- void [forward](#) ()

### Public Attributes

- Loss & [loss\\_](#)
- T [learning\\_rate\\_](#)
- T [momentum\\_](#)

## 6.8.1 Member Typedef Documentation

### 6.8.1.1 tensor\_type

```
template<typename Loss , typename T >
typedef tensor< T > ceras::gradient\_descent< Loss, T >::tensor\_type
```

## 6.8.2 Constructor & Destructor Documentation

### 6.8.2.1 gradient\_descent()

```
template<typename Loss , typename T >
ceras::gradient\_descent< Loss, T >::gradient\_descent (
    Loss & loss,
    std::size_t batch_size,
    T learning_rate = 1.0e-3,
    T momentum = 0.0 ) [inline], [noexcept]
```

## 6.8.3 Member Function Documentation

### 6.8.3.1 forward()

```
template<typename Loss , typename T >
void ceras::gradient_descent< Loss, T >::forward ( ) [inline]
```

## 6.8.4 Member Data Documentation

### 6.8.4.1 learning\_rate\_

```
template<typename Loss , typename T >
T ceras::gradient_descent< Loss, T >::learning_rate_
```

### 6.8.4.2 loss\_

```
template<typename Loss , typename T >
Loss& ceras::gradient_descent< Loss, T >::loss_
```

### 6.8.4.3 momentum\_

```
template<typename Loss , typename T >
T ceras::gradient_descent< Loss, T >::momentum_
```

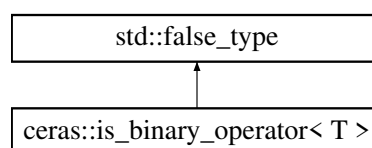
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/optimizer.hpp>

## 6.9 ceras::is\_binary\_operator< T > Struct Template Reference

```
#include <operation.hpp>
```

Inheritance diagram for ceras::is\_binary\_operator< T >:



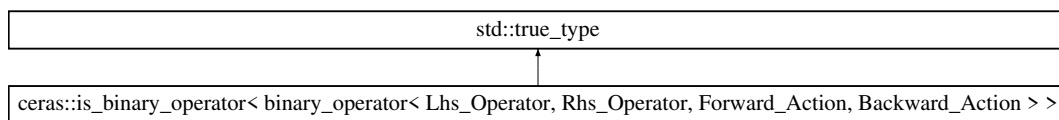
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/operation.hpp>

## 6.10 `ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > >` Struct Template Reference

```
#include <operation.hpp>
```

Inheritance diagram for `ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > >`:



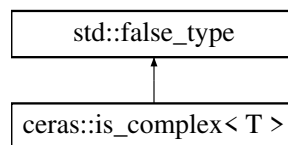
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/operation.hpp>

## 6.11 `ceras::is_complex< T >` Struct Template Reference

```
#include <complex_operator.hpp>
```

Inheritance diagram for `ceras::is_complex< T >`:



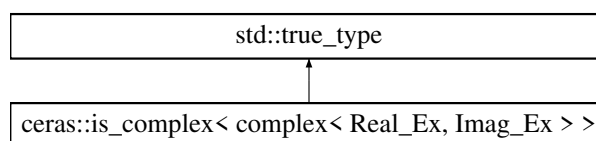
The documentation for this struct was generated from the following file:

- [/home/feng/workspace/github.repo/ceras/include/complex\\_operator.hpp](/home/feng/workspace/github.repo/ceras/include/complex_operator.hpp)

## 6.12 `ceras::is_complex< complex< Real_Ex, Imag_Ex > >` Struct Template Reference

```
#include <complex_operator.hpp>
```

Inheritance diagram for `ceras::is_complex< complex< Real_Ex, Imag_Ex > >`:



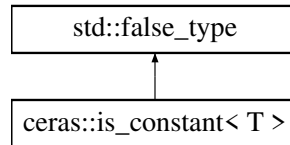
The documentation for this struct was generated from the following file:

- [/home/feng/workspace/github.repo/ceras/include/complex\\_operator.hpp](/home/feng/workspace/github.repo/ceras/include/complex_operator.hpp)

## 6.13 ceras::is\_constant< T > Struct Template Reference

```
#include <constant.hpp>
```

Inheritance diagram for ceras::is\_constant< T >:



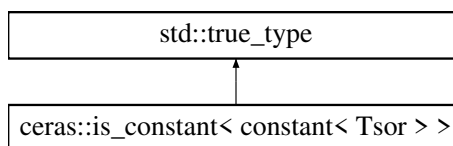
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/constant.hpp>

## 6.14 ceras::is\_constant< constant< Tsor > > Struct Template Reference

```
#include <constant.hpp>
```

Inheritance diagram for ceras::is\_constant< constant< Tsor > >:



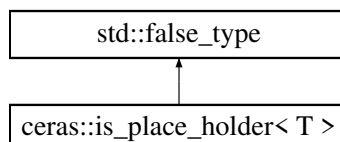
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/constant.hpp>

## 6.15 ceras::is\_place\_holder< T > Struct Template Reference

```
#include <place_holder.hpp>
```

Inheritance diagram for ceras::is\_place\_holder< T >:



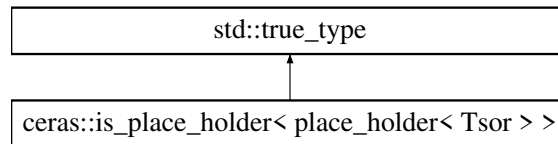
The documentation for this struct was generated from the following file:

- [/home/feng/workspace/github.repo/ceras/include/place\\_holder.hpp](/home/feng/workspace/github.repo/ceras/include/place_holder.hpp)

## 6.16 `ceras::is_place_holder< place_holder< Tsor > >` Struct Template Reference

```
#include <place_holder.hpp>
```

Inheritance diagram for `ceras::is_place_holder< place_holder< Tsor > >`:



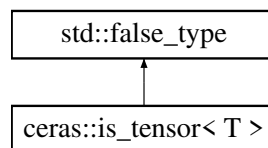
The documentation for this struct was generated from the following file:

- [/home/feng/workspace/github.repo/ceras/include/place\\_holder.hpp](/home/feng/workspace/github.repo/ceras/include/place_holder.hpp)

## 6.17 `ceras::is_tensor< T >` Struct Template Reference

```
#include <tensor.hpp>
```

Inheritance diagram for `ceras::is_tensor< T >`:



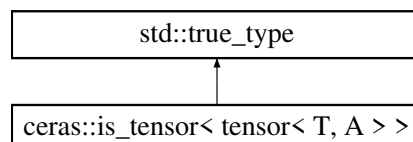
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/tensor.hpp>

## 6.18 `ceras::is_tensor< tensor< T, A > >` Struct Template Reference

```
#include <tensor.hpp>
```

Inheritance diagram for `ceras::is_tensor< tensor< T, A > >`:



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

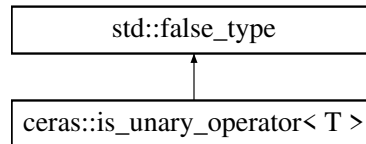
- </home/feng/workspace/github.repo/ceras/include/tensor.hpp>



## 6.19 ceras::is\_unary\_operator< T > Struct Template Reference

```
#include <operation.hpp>
```

Inheritance diagram for ceras::is\_unary\_operator< T >:



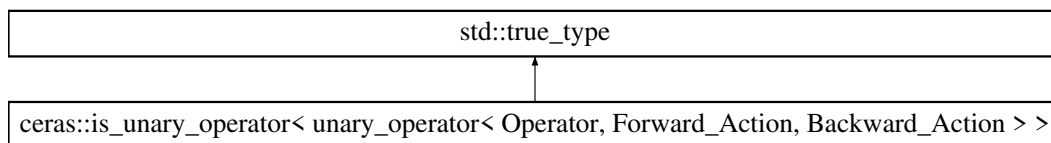
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/operation.hpp>

## 6.20 ceras::is\_unary\_operator< unary\_operator< Operator, Forward\_Action, Backward\_Action > > Struct Template Reference

```
#include <operation.hpp>
```

Inheritance diagram for ceras::is\_unary\_operator< unary\_operator< Operator, Forward\_Action, Backward\_Action > >:



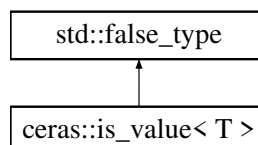
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/operation.hpp>

## 6.21 ceras::is\_value< T > Struct Template Reference

```
#include <value.hpp>
```

Inheritance diagram for ceras::is\_value< T >:



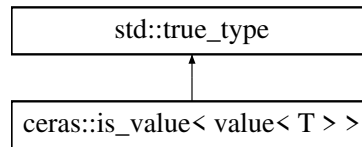
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/value.hpp>

## 6.22 `ceras::is_value< value< T > >` Struct Template Reference

```
#include <value.hpp>
```

Inheritance diagram for `ceras::is_value< value< T > >`:



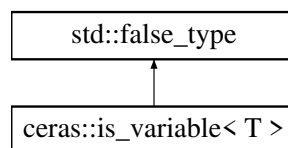
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/value.hpp>

## 6.23 `ceras::is_variable< T >` Struct Template Reference

```
#include <variable.hpp>
```

Inheritance diagram for `ceras::is_variable< T >`:



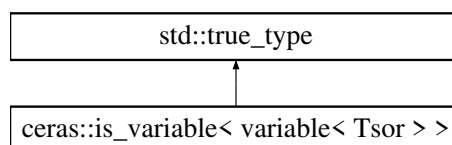
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/variable.hpp>

## 6.24 `ceras::is_variable< variable< Tsor > >` Struct Template Reference

```
#include <variable.hpp>
```

Inheritance diagram for `ceras::is_variable< variable< Tsor > >`:



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

- </home/feng/workspace/github.repo/ceras/include/variable.hpp>

## 6.25 ceras::model< Ex, Ph > Struct Template Reference

```
#include <model.hpp>
```

### Public Types

- typedef Ph [input\\_layer\\_type](#)
- typedef Ex [output\\_layer\\_type](#)

### Public Member Functions

- [input\\_layer\\_type](#) [input](#) () const noexcept
- [output\\_layer\\_type](#) [output](#) () const noexcept
- [model](#) ([input\\_layer\\_type](#) const &[place\\_holder](#), [output\\_layer\\_type](#) const &expression)
- template<Tensor Tsor>  
auto [predict](#) (Tsor const &input\_tensor)
- template<Expression Exp>  
auto [operator](#)() (Exp const &ex) const noexcept
- template<typename Loss , typename Optimizer >  
auto [compile](#) (Loss const &l, Optimizer const &o)
- void [trainable](#) (bool t)
- void [save\\_weights](#) (std::string const &file)
- void [load\\_weights](#) (std::string const &file)
- void [summary](#) (std::string const &file\_name=std::string{}) const noexcept

### Public Attributes

- [output\\_layer\\_type](#) [expression\\_](#)  
*output layer of the model.*
- [input\\_layer\\_type](#) [place\\_holder\\_](#)

#### 6.25.1 Detailed Description

```
template<Expression Ex, Place_Holder Ph>
struct ceras::model< Ex, Ph >
```

Groups an input layer (a place holder) and an output layer (an expression template) into an object.

#### Template Parameters

<i>Ex</i>	The expression template for the output layer.
<i>Ph</i>	The place holder expression for the input layer

#### 6.25.2 Member Typedef Documentation

### 6.25.2.1 input\_layer\_type

```
template<Expression Ex, Place_Holder Ph>
typedef Ph ceras::model< Ex, Ph >::input_layer_type
```

### 6.25.2.2 output\_layer\_type

```
template<Expression Ex, Place_Holder Ph>
typedef Ex ceras::model< Ex, Ph >::output_layer_type
```

## 6.25.3 Constructor & Destructor Documentation

### 6.25.3.1 model()

```
template<Expression Ex, Place_Holder Ph>
ceras::model< Ex, Ph >::model (
    input_layer_type const & place_holder,
    output_layer_type const & expression ) [inline]
```

#### Parameters

<a href="#">place_holder</a>	The input layer of the model, a place holder.
<a href="#">expression</a>	The output layer of the model, a expression template.

Example code to generate a model:

```
auto input = Input();
auto l1 = relu( Dense( 1024, 28*28 )( input ) );
auto output = sigmoid( Dense( 10, 1024 )( l1 ) );
auto m = model{ input, output };
```

## 6.25.4 Member Function Documentation

### 6.25.4.1 compile()

```
template<Expression Ex, Place_Holder Ph>
template<typename Loss , typename Optimizer >
auto ceras::model< Ex, Ph >::compile (
    Loss const & l,
    Optimizer const & o ) [inline]
```

Compile the model for training

## Parameters

<i>l</i>	The loss to minimize.
<i>o</i>	The optimizer to do the optimization.

## Returns

An instance of [compiled\\_model](#).

## Example useage:

```
model m{ ... };
unsigned long batch_size = 16;
float learning_rate = 0.001f;
auto cm = m.compile( MeanSquaredError(), SGD( batch_size, learning_rate ) );
```

## 6.25.4.2 input()

```
template<Expression Ex, Place_Holder Ph>
input_layer_type ceras::model< Ex, Ph >::input ( ) const [inline], [noexcept]
```

Returns the input layer of the model, which is a [place\\_holder](#).

## 6.25.4.3 load\_weights()

```
template<Expression Ex, Place_Holder Ph>
void ceras::model< Ex, Ph >::load_weights (
    std::string const & file ) [inline]
```

Loads all variables from a file

## 6.25.4.4 operator()()

```
template<Expression Ex, Place_Holder Ph>
template<Expression Exp>
auto ceras::model< Ex, Ph >::operator() (
    Exp const & ex ) const [inline], [noexcept]
```

Generating a new expression by using the current model.

## Parameters

<i>ex</i>	An expression that represents the input to the model.
-----------	---

## Returns

An expression that replacing the input node with a new epxression.

## Example code

```

auto x = Input(); // input, (28*28,)
auto y = Dense( 128, 28*28 )( x );
auto m1 = model( x, y ); // this model is [(28*28,) -> (128,)]
auto u = Input(); // new input, (32,)
auto v = Dense( 28*28, 32 )( u );
auto m2 = model( u, v );
auto input = Input(); // (32, )
auto ouptut = m1( m2( input ) ); // this new expression is [(32,) -> (28*28,) -> (128,)], note x is not in
    this expression any more
auto m = model( input, output ); // create a new model

```

#### 6.25.4.5 output()

```

template<Expression Ex, Place_Holder Ph>
output_layer_type ceras::model< Ex, Ph >::output ( ) const [inline], [noexcept]

```

Returns the output layer of the model.

#### 6.25.4.6 predict()

```

template<Expression Ex, Place_Holder Ph>
template<Tensor Tsor>
auto ceras::model< Ex, Ph >::predict (
    Tsor const & input_tensor ) [inline]

```

Making prediction by binding the nput data to the place\_holder\_ and evaluating expression\_.

##### Parameters

<i>input_tensor</i>	The input samples.
---------------------	--------------------

##### Returns

The result this model predicts.

##### Example to predict

```

auto input = Input();
auto l1 = relu( Dense( 1024, 28*28 )( input ) );
auto output = sigmoid( Dense( 10, 1024 )( l1 ) );
// ... train the model after defining a loss and an optimizer
auto m = model{ input, output };
auto test_data = random( {128, 28*28} ); // batch size is 128
auto result = model.predict( test_data ); // should produce an tensor of (128, 10)

```

#### 6.25.4.7 save\_weights()

```

template<Expression Ex, Place_Holder Ph>
void ceras::model< Ex, Ph >::save_weights (
    std::string const & file ) [inline]

```

Writes all variables to a file

#### 6.25.4.8 summary()

```
template<Expression Ex, Place_Holder Ph>
void ceras::model< Ex, Ph >::summary (
    std::string const & file_name = std::string{} ) const [inline], [noexcept]
```

Print the model summary to console or to a file.

##### Parameters

<code>file_name</code>	The file to save the summary. If empty, the summary will be printed to console. Empty by default.
------------------------	---

#### 6.25.4.9 trainable()

```
template<Expression Ex, Place_Holder Ph>
void ceras::model< Ex, Ph >::trainable (
    bool t ) [inline]
```

### 6.25.5 Member Data Documentation

#### 6.25.5.1 expression\_

```
template<Expression Ex, Place_Holder Ph>
output_layer_type ceras::model< Ex, Ph >::expression_
```

output layer of the model.

#### 6.25.5.2 place\_holder\_

```
template<Expression Ex, Place_Holder Ph>
input_layer_type ceras::model< Ex, Ph >::place_holder_
```

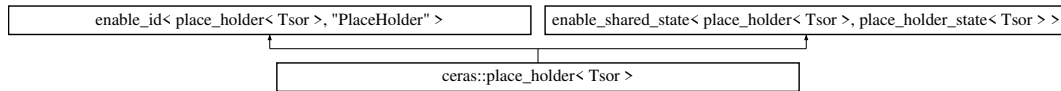
The documentation for this struct was generated from the following file:

- `/home/feng/workspace/github.repo/ceras/include/model.hpp`

## 6.26 ceras::place\_holder< Tsr > Struct Template Reference

```
#include <place_holder.hpp>
```

Inheritance diagram for ceras::place\_holder< Tsr >:



### Public Types

- typedef Tsr [tensor\\_type](#)

### Public Member Functions

- [place\\_holder](#) ([place\\_holder](#) const &other)=default
- [place\\_holder](#) ([place\\_holder](#) &&other)=default
- [place\\_holder](#) & [operator=](#) ([place\\_holder](#) const &other)=default
- [place\\_holder](#) & [operator=](#) ([place\\_holder](#) &&other)=default
- [place\\_holder](#) ()
- [place\\_holder](#) (std::vector< unsigned long > const &shape\_hint)
- void [bind](#) (Tsr data)
- Tsr const [forward](#) () const
- void [reset](#) ()
- void [backward](#) (auto) const noexcept

### 6.26.1 Member Typedef Documentation

#### 6.26.1.1 tensor\_type

```
template<Tensor Tsr>
typedef Tsr ceras::place\_holder< Tsr >::tensor\_type
```

### 6.26.2 Constructor & Destructor Documentation

#### 6.26.2.1 place\_holder() [1/4]

```
template<Tensor Tsr>
ceras::place\_holder< Tsr >::place\_holder (
    place\_holder< Tsr > const & other ) [default]
```



### 6.26.2.2 place\_holder() [2/4]

```
template<Tensor Tsor>
ceras::place_holder< Tsor >::place_holder (
    place_holder< Tsor > && other ) [default]
```

### 6.26.2.3 place\_holder() [3/4]

```
template<Tensor Tsor>
ceras::place_holder< Tsor >::place_holder ( ) [inline]
```

### 6.26.2.4 place\_holder() [4/4]

```
template<Tensor Tsor>
ceras::place_holder< Tsor >::place_holder (
    std::vector< unsigned long > const & shape_hint ) [inline]
```

## 6.26.3 Member Function Documentation

### 6.26.3.1 backward()

```
template<Tensor Tsor>
void ceras::place_holder< Tsor >::backward (
    auto ) const [inline], [noexcept]
```

### 6.26.3.2 bind()

```
template<Tensor Tsor>
void ceras::place_holder< Tsor >::bind (
    Tsor data ) [inline]
```

### 6.26.3.3 forward()

```
template<Tensor Tsor>
Tsor const ceras::place_holder< Tsor >::forward ( ) const [inline]
```

**6.26.3.4 operator=() [1/2]**

```
template<Tensor Tsor>
place_holder& ceras::place_holder< Tsor >::operator= (
    place_holder< Tsor > && other ) [default]
```

**6.26.3.5 operator=() [2/2]**

```
template<Tensor Tsor>
place_holder& ceras::place_holder< Tsor >::operator= (
    place_holder< Tsor > const & other ) [default]
```

**6.26.3.6 reset()**

```
template<Tensor Tsor>
void ceras::place_holder< Tsor >::reset ( ) [inline]
```

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

- [/home/feng/workspace/github.repo/ceras/include/place\\_holder.hpp](/home/feng/workspace/github.repo/ceras/include/place_holder.hpp)

**6.27 ceras::place\_holder\_state< Tsor > Struct Template Reference**

```
#include <place_holder.hpp>
```

**Public Attributes**

- Tsor [data\\_](#)
- std::vector< unsigned long > [shape\\_hint\\_](#)

**6.27.1 Member Data Documentation****6.27.1.1 data\_**

```
template<Tensor Tsor>
Tsor ceras::place_holder_state< Tsor >::data_
```

### 6.27.1.2 shape\_hint\_

```
template<Tensor Tsor>
std::vector< unsigned long> ceras::place_holder_state< Tsor >::shape_hint_
```

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

- [/home/feng/workspace/github.repo/ceras/include/place\\_holder.hpp](#)

## 6.28 ceras::regularizer< Float > Struct Template Reference

```
#include <variable.hpp>
```

### Public Types

- typedef Float [value\\_type](#)

### Public Member Functions

- constexpr [regularizer](#) ([value\\_type](#) l1, [value\\_type](#) l2, bool synchronized) noexcept

### Public Attributes

- [value\\_type](#) l1\_
- [value\\_type](#) l2\_
- bool [synchronized\\_](#)

## 6.28.1 Member Typedef Documentation

### 6.28.1.1 value\_type

```
template<typename Float >
typedef Float ceras::regularizer< Float >::value_type
```

## 6.28.2 Constructor & Destructor Documentation

### 6.28.2.1 regularizer()

```
template<typename Float >
constexpr ceras::regularizer< Float >::regularizer (
    value_type l1,
    value_type l2,
    bool synchronized ) [inline], [constexpr], [noexcept]
```

## 6.28.3 Member Data Documentation

### 6.28.3.1 l1\_

```
template<typename Float >
value_type ceras::regularizer< Float >::l1_
```

### 6.28.3.2 l2\_

```
template<typename Float >
value_type ceras::regularizer< Float >::l2_
```

### 6.28.3.3 synchronized\_

```
template<typename Float >
bool ceras::regularizer< Float >::synchronized_
```

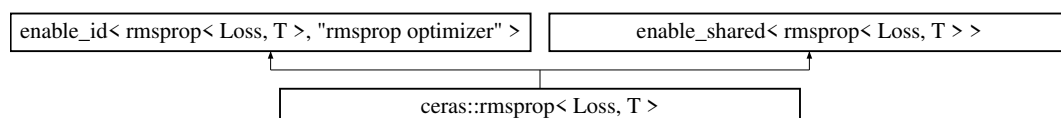
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/variable.hpp>

## 6.29 ceras::rmsprop< Loss, T > Struct Template Reference

```
#include <optimizer.hpp>
```

Inheritance diagram for ceras::rmsprop< Loss, T >:



## Public Types

- typedef [tensor](#)< T > [tensor\\_type](#)

## Public Member Functions

- [rmsprop](#) (Loss &loss, std::size\_t batch\_size, T learning\_rate=1.0e-1, T rho=0.9, T decay=0.0) noexcept
- void [forward](#) ()

## Public Attributes

- Loss & [loss\\_](#)
- T [learning\\_rate\\_](#)
- T [rho\\_](#)
- T [decay\\_](#)
- unsigned long [iterations\\_](#)

### 6.29.1 Member Typedef Documentation

#### 6.29.1.1 tensor\_type

```
template<typename Loss , typename T >
typedef tensor< T > ceras::rmsprop< Loss, T >::tensor\_type
```

### 6.29.2 Constructor & Destructor Documentation

#### 6.29.2.1 rmsprop()

```
template<typename Loss , typename T >
ceras::rmsprop< Loss, T >::rmsprop (
    Loss & loss,
    std::size_t batch_size,
    T learning_rate = 1.0e-1,
    T rho = 0.9,
    T decay = 0.0 ) [inline], [noexcept]
```

### 6.29.3 Member Function Documentation

### 6.29.3.1 forward()

```
template<typename Loss , typename T >
void ceras::rmsprop< Loss, T >::forward ( ) [inline]
```

## 6.29.4 Member Data Documentation

### 6.29.4.1 decay\_

```
template<typename Loss , typename T >
T ceras::rmsprop< Loss, T >::decay_
```

### 6.29.4.2 iterations\_

```
template<typename Loss , typename T >
unsigned long ceras::rmsprop< Loss, T >::iterations_
```

### 6.29.4.3 learning\_rate\_

```
template<typename Loss , typename T >
T ceras::rmsprop< Loss, T >::learning_rate_
```

### 6.29.4.4 loss\_

```
template<typename Loss , typename T >
Loss& ceras::rmsprop< Loss, T >::loss_
```

### 6.29.4.5 rho\_

```
template<typename Loss , typename T >
T ceras::rmsprop< Loss, T >::rho_
```

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

- [/home/feng/workspace/github.repo/ceras/include/optimizer.hpp](#)

## 6.30 ceras::ceras\_private::session< Tzor > Struct Template Reference

```
#include <session.hpp>
```

### Public Types

- typedef [place\\_holder](#)< Tzor > [place\\_holder\\_type](#)
- typedef [variable](#)< Tzor > [variable\\_type](#)
- typedef [variable\\_state](#)< Tzor > [variable\\_state\\_type](#)

### Public Member Functions

- [session](#) ()
- [session](#) ([session](#) const &)=delete
- [session](#) ([session](#) &&)=default
- [session](#) & [operator=](#) ([session](#) const &)=delete
- [session](#) & [operator=](#) ([session](#) &&)=default
- void [rebind](#) ([place\\_holder\\_type](#) &p\_holder, Tzor const &[value](#))
- void [bind](#) ([place\\_holder\\_type](#) &p\_holder, Tzor const &[value](#))
- void [remember](#) ([variable\\_type](#) const &[v](#))
- template<typename Operation >  
  auto [run](#) (Operation &op) const
- template<typename Operation >  
  void [tap](#) (Operation &op) const
- void [deserialize](#) (std::string const &file\_path)
- void [serialize](#) (std::string const &file\_path) const
- void [save](#) (std::string const &file\_path) const
- void [restore](#) (std::string const &file\_path)
- [~session](#) ()

### Public Attributes

- std::vector< [place\\_holder\\_type](#) > [place\\_holders\\_](#)
- std::map< int, [variable\\_type](#) > [variables\\_](#)

### 6.30.1 Member Typedef Documentation

#### 6.30.1.1 place\_holder\_type

```
template<Tensor Tzor>
typedef place\_holder<Tzor> ceras::ceras\_private::session< Tzor >::place\_holder\_type
```

### 6.30.1.2 variable\_state\_type

```
template<Tensor Tsor>
typedef variable_state<Tsor> ceras::ceras_private::session< Tsor >::variable_state_type
```

### 6.30.1.3 variable\_type

```
template<Tensor Tsor>
typedef variable<Tsor> ceras::ceras_private::session< Tsor >::variable_type
```

## 6.30.2 Constructor & Destructor Documentation

### 6.30.2.1 session() [1/3]

```
template<Tensor Tsor>
ceras::ceras_private::session< Tsor >::session ( ) [inline]
```

### 6.30.2.2 session() [2/3]

```
template<Tensor Tsor>
ceras::ceras_private::session< Tsor >::session (
    session< Tsor > const & ) [delete]
```

### 6.30.2.3 session() [3/3]

```
template<Tensor Tsor>
ceras::ceras_private::session< Tsor >::session (
    session< Tsor > && ) [default]
```

### 6.30.2.4 ~session()

```
template<Tensor Tsor>
ceras::ceras_private::session< Tsor >::~~session ( ) [inline]
```



## 6.30.3 Member Function Documentation

### 6.30.3.1 bind()

```
template<Tensor Tsor>
void ceras::ceras_private::session< Tsor >::bind (
    place_holder_type & p_holder,
    Tsor const & value ) [inline]
```

### 6.30.3.2 deserialize()

```
template<Tensor Tsor>
void ceras::ceras_private::session< Tsor >::deserialize (
    std::string const & file_path ) [inline]
```

### 6.30.3.3 operator=() [1/2]

```
template<Tensor Tsor>
session& ceras::ceras_private::session< Tsor >::operator= (
    session< Tsor > && ) [default]
```

### 6.30.3.4 operator=() [2/2]

```
template<Tensor Tsor>
session& ceras::ceras_private::session< Tsor >::operator= (
    session< Tsor > const & ) [delete]
```

### 6.30.3.5 rebind()

```
template<Tensor Tsor>
void ceras::ceras_private::session< Tsor >::rebind (
    place_holder_type & p_holder,
    Tsor const & value ) [inline]
```

#### 6.30.3.6 remember()

```
template<Tensor Tsor>
void ceras::ceras_private::session< Tsor >::remember (
    variable_type const & v ) [inline]
```

#### 6.30.3.7 restore()

```
template<Tensor Tsor>
void ceras::ceras_private::session< Tsor >::restore (
    std::string const & file_path ) [inline]
```

#### 6.30.3.8 run()

```
template<Tensor Tsor>
template<typename Operation >
auto ceras::ceras_private::session< Tsor >::run (
    Operation & op ) const [inline]
```

#### 6.30.3.9 save()

```
template<Tensor Tsor>
void ceras::ceras_private::session< Tsor >::save (
    std::string const & file_path ) const [inline]
```

#### 6.30.3.10 serialize()

```
template<Tensor Tsor>
void ceras::ceras_private::session< Tsor >::serialize (
    std::string const & file_path ) const [inline]
```

#### 6.30.3.11 tap()

```
template<Tensor Tsor>
template<typename Operation >
void ceras::ceras_private::session< Tsor >::tap (
    Operation & op ) const [inline]
```

### 6.30.4 Member Data Documentation

#### 6.30.4.1 place\_holders\_

```
template<Tensor Tsor>
std::vector<place_holder_type> ceras::ceras_private::session< Tsor >::place_holders_
```

#### 6.30.4.2 variables\_

```
template<Tensor Tsor>
std::map<int, variable_type> ceras::ceras_private::session< Tsor >::variables_
```

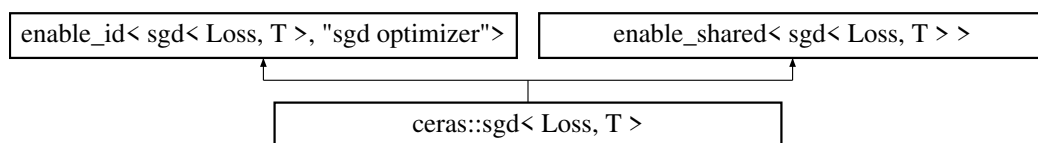
The documentation for this struct was generated from the following file:

- /home/feng/workspace/github.repo/ceras/include/session.hpp

## 6.31 ceras::sgd< Loss, T > Struct Template Reference

```
#include <optimizer.hpp>
```

Inheritance diagram for ceras::sgd< Loss, T >:



### Public Types

- typedef `tensor< T >` `tensor_type`

### Public Member Functions

- `sgd` (Loss &loss, std::size\_t batch\_size, T learning\_rate=1.0e-1, T momentum=0.0, T decay=0.0, bool nestorov=false) noexcept
- void `forward` ()

## Public Attributes

- Loss & [loss\\_](#)
- T [learning\\_rate\\_](#)
- T [momentum\\_](#)
- T [decay\\_](#)
- bool [nesterov\\_](#)
- unsigned long [iterations\\_](#)

## 6.31.1 Member Typedef Documentation

### 6.31.1.1 [tensor\\_type](#)

```
template<typename Loss , typename T >
typedef tensor< T > ceras::sgd< Loss, T >::tensor\_type
```

## 6.31.2 Constructor & Destructor Documentation

### 6.31.2.1 [sgd\(\)](#)

```
template<typename Loss , typename T >
ceras::sgd< Loss, T >::sgd (
    Loss & loss,
    std::size_t batch\_size,
    T learning\_rate = 1.0e-1,
    T momentum = 0.0,
    T decay = 0.0,
    bool nesterov = false ) [inline], [noexcept]
```

## 6.31.3 Member Function Documentation

### 6.31.3.1 [forward\(\)](#)

```
template<typename Loss , typename T >
void ceras::sgd< Loss, T >::forward ( ) [inline]
```

## 6.31.4 Member Data Documentation

#### 6.31.4.1 decay\_

```
template<typename Loss , typename T >  
T ceras::sgd< Loss, T >::decay_
```

#### 6.31.4.2 iterations\_

```
template<typename Loss , typename T >  
unsigned long ceras::sgd< Loss, T >::iterations_
```

#### 6.31.4.3 learning\_rate\_

```
template<typename Loss , typename T >  
T ceras::sgd< Loss, T >::learning_rate_
```

#### 6.31.4.4 loss\_

```
template<typename Loss , typename T >  
Loss& ceras::sgd< Loss, T >::loss_
```

#### 6.31.4.5 momentum\_

```
template<typename Loss , typename T >  
T ceras::sgd< Loss, T >::momentum_
```

#### 6.31.4.6 nesterov\_

```
template<typename Loss , typename T >  
bool ceras::sgd< Loss, T >::nesterov_
```

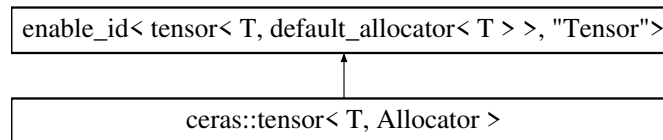
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/optimizer.hpp>

## 6.32 ceras::tensor< T, Allocator > Struct Template Reference

```
#include <tensor.hpp>
```

Inheritance diagram for ceras::tensor< T, Allocator >:



### Public Types

- typedef T [value\\_type](#)
- typedef Allocator [allocator](#)
- typedef std::vector< T, Allocator > [vector\\_type](#)
- typedef std::shared\_ptr< [vector\\_type](#) > [shared\\_vector](#)
- typedef [tensor](#) [self\\_type](#)

### Public Member Functions

- constexpr auto [begin](#) () noexcept
- constexpr auto [begin](#) () const noexcept
- constexpr auto [cbegin](#) () const noexcept
- constexpr auto [end](#) () noexcept
- constexpr auto [end](#) () const noexcept
- constexpr auto [cend](#) () const noexcept
- constexpr [self\\_type](#) & [reset](#) (T val=T{0})
- constexpr unsigned long [ndim](#) () const noexcept
- constexpr [self\\_type](#) & [deep\\_copy](#) ([self\\_type](#) const &other)
- constexpr [self\\_type](#) const [deep\\_copy](#) () const
- constexpr [self\\_type](#) const [copy](#) () const
- constexpr [value\\_type](#) & [operator\[\]](#) (unsigned long idx)
- constexpr [value\\_type](#) const & [operator\[\]](#) (unsigned long idx) const
- [tensor](#) ()
- constexpr [tensor](#) (std::vector< unsigned long > const &[shape](#), std::initializer\_list< T > init, const Allocator &alloc=Allocator())
- constexpr [tensor](#) (std::vector< unsigned long > const &[shape](#))
- constexpr [tensor](#) (std::vector< unsigned long > const &[shape](#), T init)
- constexpr [tensor](#) ([tensor](#) const &other, unsigned long memory\_offset)
- constexpr [tensor](#) ([self\\_type](#) const &other) noexcept
- constexpr [tensor](#) ([self\\_type](#) &&other) noexcept
- constexpr [self\\_type](#) & [operator=](#) ([self\\_type](#) const &other) noexcept
- constexpr [self\\_type](#) & [operator=](#) ([self\\_type](#) &&other) noexcept
- constexpr std::vector< unsigned long > const & [shape](#) () const noexcept
- constexpr unsigned long [size](#) () const noexcept
- constexpr [self\\_type](#) & [resize](#) (std::vector< unsigned long > const &new\_shape)
- constexpr [self\\_type](#) & [reshape](#) (std::vector< unsigned long > const &new\_shape)
- constexpr [self\\_type](#) & [shrink\\_to](#) (std::vector< unsigned long > const &new\_shape)
- constexpr [self\\_type](#) & [creep\\_to](#) (unsigned long new\_memory\_offset)
- constexpr bool [empty](#) () const noexcept

- constexpr [value\\_type](#) \* [data](#) () noexcept
- constexpr const [value\\_type](#) \* [data](#) () const noexcept
- template<typename Function >  
constexpr [self\\_type](#) & [map](#) (Function const &f)
- constexpr [self\\_type](#) & [operator+=](#) ([self\\_type](#) const &other)
- constexpr [self\\_type](#) & [operator+=](#) ([value\\_type](#) x)
- constexpr [self\\_type](#) & [operator-=](#) ([self\\_type](#) const &other)
- constexpr [self\\_type](#) & [operator-=](#) ([value\\_type](#) x)
- constexpr [self\\_type](#) & [operator\\*=](#) ([self\\_type](#) const &other)
- constexpr [self\\_type](#) & [operator\\*=](#) ([value\\_type](#) x)
- constexpr [self\\_type](#) & [operator/=](#) ([self\\_type](#) const &other)
- constexpr [self\\_type](#) & [operator/=](#) ([value\\_type](#) x)
- constexpr [self\\_type](#) const [operator-](#) () const
- constexpr [value\\_type as\\_scalar](#) () const noexcept
- template<typename U >  
constexpr auto [as\\_type](#) () const noexcept
- [tensor slice](#) (unsigned long m, unsigned long n) const noexcept

## Public Attributes

- std::vector< unsigned long > [shape\\_](#)
- unsigned long [memory\\_offset\\_](#)
- [shared\\_vector](#) [vector\\_](#)

## 6.32.1 Member Typedef Documentation

### 6.32.1.1 allocator

```
template<typename T , typename Allocator = default_allocator<T>>
typedef Allocator ceras::tensor< T, Allocator >::allocator
```

### 6.32.1.2 self\_type

```
template<typename T , typename Allocator = default_allocator<T>>
typedef tensor ceras::tensor< T, Allocator >::self\_type
```

### 6.32.1.3 shared\_vector

```
template<typename T , typename Allocator = default_allocator<T>>
typedef std::shared_ptr<vector\_type> ceras::tensor< T, Allocator >::shared\_vector
```

#### 6.32.1.4 value\_type

```
template<typename T , typename Allocator = default_allocator<T>>
typedef T ceras::tensor< T, Allocator >::value\_type
```

#### 6.32.1.5 vector\_type

```
template<typename T , typename Allocator = default_allocator<T>>
typedef std::vector<T, Allocator> ceras::tensor< T, Allocator >::vector\_type
```

### 6.32.2 Constructor & Destructor Documentation

#### 6.32.2.1 tensor() [1/7]

```
template<typename T , typename Allocator = default_allocator<T>>
ceras::tensor< T, Allocator >::tensor ( ) [inline]
```

#### 6.32.2.2 tensor() [2/7]

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr ceras::tensor< T, Allocator >::tensor (
    std::vector< unsigned long > const & shape,
    std::initializer_list< T > init,
    const Allocator & alloc = Allocator() ) [inline], [constexpr]
```

#### 6.32.2.3 tensor() [3/7]

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr ceras::tensor< T, Allocator >::tensor (
    std::vector< unsigned long > const & shape ) [inline], [constexpr]
```

#### 6.32.2.4 tensor() [4/7]

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr ceras::tensor< T, Allocator >::tensor (
    std::vector< unsigned long > const & shape,
    T init ) [inline], [constexpr]
```



**6.32.2.5 tensor() [5/7]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr ceras::tensor< T, Allocator >::tensor (
    tensor< T, Allocator > const & other,
    unsigned long memory_offset ) [inline], [constexpr]
```

**6.32.2.6 tensor() [6/7]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr ceras::tensor< T, Allocator >::tensor (
    self_type const & other ) [inline], [constexpr], [noexcept]
```

**6.32.2.7 tensor() [7/7]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr ceras::tensor< T, Allocator >::tensor (
    self_type && other ) [inline], [constexpr], [noexcept]
```

**6.32.3 Member Function Documentation****6.32.3.1 as\_scalar()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr value_type ceras::tensor< T, Allocator >::as_scalar ( ) const [inline], [constexpr],
[noexcept]
```

**6.32.3.2 as\_type()**

```
template<typename T , typename Allocator = default_allocator<T>>
template<typename U >
constexpr auto ceras::tensor< T, Allocator >::as_type ( ) const [inline], [constexpr], [noexcept]
```

**6.32.3.3 begin() [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr auto ceras::tensor< T, Allocator >::begin ( ) const [inline], [constexpr], [noexcept]
```

**6.32.3.4 begin() [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr auto ceras::tensor< T, Allocator >::begin ( ) [inline], [constexpr], [noexcept]
```

**6.32.3.5 cbegin()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr auto ceras::tensor< T, Allocator >::cbegin ( ) const [inline], [constexpr], [noexcept]
```

**6.32.3.6 cend()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr auto ceras::tensor< T, Allocator >::cend ( ) const [inline], [constexpr], [noexcept]
```

**6.32.3.7 copy()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type const ceras::tensor< T, Allocator >::copy ( ) const [inline], [constexpr]
```

**6.32.3.8 creep\_to()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::creep_to (
    unsigned long new_memory_offset ) [inline], [constexpr]
```

**6.32.3.9 data() [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr const value_type* ceras::tensor< T, Allocator >::data ( ) const [inline], [constexpr],
[noexcept]
```

**6.32.3.10 data() [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr value_type* ceras::tensor< T, Allocator >::data ( ) [inline], [constexpr], [noexcept]
```

**6.32.3.11 deep\_copy() [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type const ceras::tensor< T, Allocator >::deep_copy ( ) const [inline], [constexpr]
```

**6.32.3.12 deep\_copy() [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::deep_copy (
    self_type const & other ) [inline], [constexpr]
```

**6.32.3.13 empty()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr bool ceras::tensor< T, Allocator >::empty ( ) const [inline], [constexpr], [noexcept]
```

**6.32.3.14 end() [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr auto ceras::tensor< T, Allocator >::end ( ) const [inline], [constexpr], [noexcept]
```

**6.32.3.15 end() [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr auto ceras::tensor< T, Allocator >::end ( ) [inline], [constexpr], [noexcept]
```

**6.32.3.16 map()**

```
template<typename T , typename Allocator = default_allocator<T>>
template<typename Function >
constexpr self_type& ceras::tensor< T, Allocator >::map (
    Function const & f ) [inline], [constexpr]
```

**6.32.3.17 ndim()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr unsigned long ceras::tensor< T, Allocator >::ndim ( ) const [inline], [constexpr],
[noexcept]
```

**6.32.3.18 operator\*=( ) [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator*= (
    self_type const & other ) [inline], [constexpr]
```

**6.32.3.19 operator\*=( ) [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator*= (
    value_type x ) [inline], [constexpr]
```

**6.32.3.20 operator+=( ) [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator+= (
    self_type const & other ) [inline], [constexpr]
```

**6.32.3.21 operator+=( ) [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator+= (
    value_type x ) [inline], [constexpr]
```

**6.32.3.22 operator-()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type const ceras::tensor< T, Allocator >::operator- ( ) const [inline], [constexpr]
```

**6.32.3.23 operator-=( ) [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator-= (
    self_type const & other ) [inline], [constexpr]
```

**6.32.3.24 operator-=( ) [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator-= (
    value_type x ) [inline], [constexpr]
```

**6.32.3.25 operator/=( ) [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator/= (
    self_type const & other ) [inline], [constexpr]
```

**6.32.3.26 operator/=( ) [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator/= (
    value_type x ) [inline], [constexpr]
```

**6.32.3.27 operator=( ) [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator= (
    self_type && other ) [inline], [constexpr], [noexcept]
```

**6.32.3.28 operator=( ) [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator= (
    self_type const & other ) [inline], [constexpr], [noexcept]
```

**6.32.3.29 operator[]() [1/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr value_type& ceras::tensor< T, Allocator >::operator[] (
    unsigned long idx ) [inline], [constexpr]
```

**6.32.3.30 operator[]() [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr value_type const& ceras::tensor< T, Allocator >::operator[] (
    unsigned long idx ) const [inline], [constexpr]
```

**6.32.3.31 reset()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::reset (
    T val = T{0} ) [inline], [constexpr]
```

Resetting all elements in the tensor to a fixed value (default to 0), without change the shape.

Example code:

```
tensor<float> ts;
//...
ts.reset();
```

**6.32.3.32 reshape()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::reshape (
    std::vector< unsigned long > const & new_shape ) [inline], [constexpr]
```

**6.32.3.33 resize()**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::resize (
    std::vector< unsigned long > const & new_shape ) [inline], [constexpr]
```

### 6.32.3.34 shape()

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr std::vector< unsigned long > const& ceras::tensor< T, Allocator >::shape ( ) const
[inline], [constexpr], [noexcept]
```

### 6.32.3.35 shrink\_to()

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::shrink_to (
    std::vector< unsigned long > const & new_shape ) [inline], [constexpr]
```

### 6.32.3.36 size()

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr unsigned long ceras::tensor< T, Allocator >::size ( ) const [inline], [constexpr],
[noexcept]
```

### 6.32.3.37 slice()

```
template<typename T , typename Allocator = default_allocator<T>>
tensor ceras::tensor< T, Allocator >::slice (
    unsigned long m,
    unsigned long n ) const [inline], [noexcept]
```

## 6.32.4 Member Data Documentation

### 6.32.4.1 memory\_offset\_

```
template<typename T , typename Allocator = default_allocator<T>>
unsigned long ceras::tensor< T, Allocator >::memory_offset_
```

### 6.32.4.2 shape\_

```
template<typename T , typename Allocator = default_allocator<T>>
std::vector<unsigned long> ceras::tensor< T, Allocator >::shape_
```

### 6.32.4.3 vector\_

```
template<typename T , typename Allocator = default_allocator<T>>
shared_vector ceras::tensor< T, Allocator >::vector_
```

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

- </home/feng/workspace/github.repo/ceras/include/tensor.hpp>

## 6.33 ceras::tensor\_deduction< L, R > Struct Template Reference

```
#include <value.hpp>
```

### Public Types

- using [op\\_type](#) = std::conditional< [is\\_value\\_v](#)< L >, R, L >::type
- using [tensor\\_type](#) = std::remove\_cv\_t< decltype(std::declval< [op\\_type](#) >()).forward())>

### 6.33.1 Member Typedef Documentation

#### 6.33.1.1 op\_type

```
template<typename L , typename R >
using ceras::tensor_deduction< L, R >::op_type = std::conditional<is_value_v<L>, R, L>::type
```

#### 6.33.1.2 tensor\_type

```
template<typename L , typename R >
using ceras::tensor_deduction< L, R >::tensor_type = std::remove_cv_t<decltype(std::declval<op_type>()).forwa
```

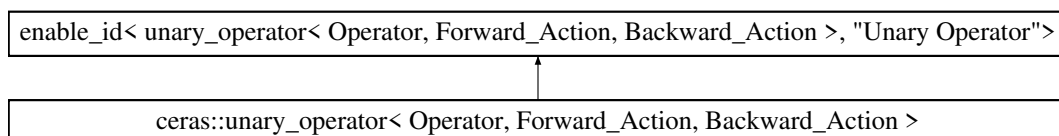
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/value.hpp>

## 6.34 ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action > Struct Template Reference

```
#include <operation.hpp>
```

Inheritance diagram for ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >:





## Public Member Functions

- [unary\\_operator](#) ([Operator](#) const &op, [Forward\\_Action](#) const &forward\_action, [Backward\\_Action](#) const &backward\_action) noexcept
- auto [forward](#) ()
- void [backward](#) ([tensor\\_type](#) const &grad)

## Public Attributes

- [Operator](#) op\_
- [Forward\\_Action](#) [forward\\_action\\_](#)
- [Backward\\_Action](#) [backward\\_action\\_](#)
- decltype(std::declval< [Forward\\_Action](#) >().std::declval< decltype([op\\_](#))>().[forward](#)() ) typedef [tensor\\_type](#)
- [tensor\\_type](#) input\_data\_
- [tensor\\_type](#) output\_data\_

## 6.34.1 Constructor & Destructor Documentation

### 6.34.1.1 unary\_operator()

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
ceras::unary_operator< Operator , Forward_Action, Backward_Action >::unary_operator (
    Operator const & op,
    Forward\_Action const & forward_action,
    Backward\_Action const & backward_action ) [inline], [noexcept]
```

## 6.34.2 Member Function Documentation

### 6.34.2.1 backward()

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
void ceras::unary_operator< Operator , Forward_Action, Backward_Action >::backward (
    tensor\_type const & grad ) [inline]
```

### 6.34.2.2 forward()

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
auto ceras::unary_operator< Operator , Forward_Action, Backward_Action >::forward ( ) [inline]
```

### 6.34.3 Member Data Documentation

#### 6.34.3.1 backward\_action\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
Backward_Action ceras::unary_operator< Operator, Forward_Action, Backward_Action >::backward_↵
_action_
```

#### 6.34.3.2 forward\_action\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
Forward_Action ceras::unary_operator< Operator, Forward_Action, Backward_Action >::forward_↵
action_
```

#### 6.34.3.3 input\_data\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
tensor_type ceras::unary_operator< Operator, Forward_Action, Backward_Action >::input_data_
```

#### 6.34.3.4 op\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
Operator ceras::unary_operator< Operator, Forward_Action, Backward_Action >::op_
```

#### 6.34.3.5 output\_data\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
tensor_type ceras::unary_operator< Operator, Forward_Action, Backward_Action >::output_data_
```

#### 6.34.3.6 tensor\_type

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
decltype( std::declval<Forward_Action>() ( std::declval<decltype(op_)>().forward() ) ) typedef
ceras::unary_operator< Operator, Forward_Action, Backward_Action >::tensor_type
```

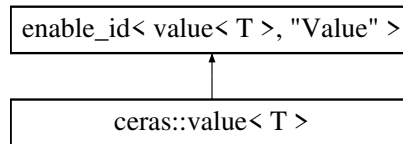
The documentation for this struct was generated from the following file:

- </home/feng/workspace/github.repo/ceras/include/operation.hpp>

## 6.35 ceras::value< T > Struct Template Reference

```
#include <value.hpp>
```

Inheritance diagram for ceras::value< T >:



### Public Types

- typedef T [value\\_type](#)

### Public Member Functions

- [value](#) ()=delete
- [value](#) ([value\\_type](#) v) noexcept
- [value](#) ([value](#) const &) noexcept=default
- [value](#) ([value](#) &&) noexcept=default
- [value](#) & [operator=](#) ([value](#) const &) noexcept=default
- [value](#) & [operator=](#) ([value](#) &&) noexcept=default
- void [backward](#) (auto) noexcept
- template<Tensor Tsor>  
Tsor const [forward](#) (Tsor const &refer) const

### Public Attributes

- [value\\_type](#) data\_

### 6.35.1 Member Typedef Documentation

#### 6.35.1.1 value\_type

```
template<typename T >
typedef T ceras::value< T >::value_type
```

### 6.35.2 Constructor & Destructor Documentation

**6.35.2.1 value() [1/4]**

```
template<typename T >
ceras::value< T >::value ( ) [delete]
```

**6.35.2.2 value() [2/4]**

```
template<typename T >
ceras::value< T >::value (
    value_type v ) [inline], [noexcept]
```

**6.35.2.3 value() [3/4]**

```
template<typename T >
ceras::value< T >::value (
    value< T > const & ) [default], [noexcept]
```

**6.35.2.4 value() [4/4]**

```
template<typename T >
ceras::value< T >::value (
    value< T > && ) [default], [noexcept]
```

**6.35.3 Member Function Documentation****6.35.3.1 backward()**

```
template<typename T >
void ceras::value< T >::backward (
    auto ) [inline], [noexcept]
```

**6.35.3.2 forward()**

```
template<typename T >
template<Tensor Tsor>
Tsor const ceras::value< T >::forward (
    Tsor const & refer ) const [inline]
```

## 6.35.3.3 operator=() [1/2]

```
template<typename T >
value& ceras::value< T >::operator= (
    value< T > && ) [default], [noexcept]
```

## 6.35.3.4 operator=() [2/2]

```
template<typename T >
value& ceras::value< T >::operator= (
    value< T > const & ) [default], [noexcept]
```

## 6.35.4 Member Data Documentation

## 6.35.4.1 data\_

```
template<typename T >
value_type ceras::value< T >::data_
```

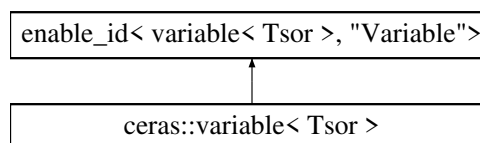
The documentation for this struct was generated from the following file:

- /home/feng/workspace/github.repo/ceras/include/value.hpp

## 6.36 ceras::variable&lt; T&gt; Struct Template Reference

```
#include <variable.hpp>
```

Inheritance diagram for ceras::variable< T>:



## Public Types

- typedef T& tensor\_type
- typedef tensor\_type::value\_type value\_type

## Public Member Functions

- `variable` (`tensor_type` const &`data`, `value_type` l1=`value_type`{0}, `value_type` l2=`value_type`{0}, bool `trainable`=true)
- `variable` ()=delete
- `variable` (`variable` const &`other`)=default
- `variable` (`variable` &&)=default
- `variable` & `operator=` (`variable` &&)=default
- `variable` & `operator=` (`variable` const &`other`)=default
- `tensor_type` const `forward` () noexcept
- void `backward` (auto const &`grad`) noexcept
- std::vector< std::size\_t > `shape` () const noexcept
- std::vector< `tensor_type` > & `contexts` ()
- std::vector< `tensor_type` > `contexts` () const
- `tensor_type` & `data` ()
- `tensor_type` `data` () const
- `tensor_type` & `gradient` ()
- `tensor_type` `gradient` () const
- void `reset` ()
- bool `trainable` () const noexcept
- void `trainable` (bool t)

## Public Attributes

- std::shared\_ptr< `variable_state`< `tensor_type` > > `state_`
- `regularizer`< `value_type` > `regularizer_`
- bool `trainable_`

## 6.36.1 Member Typedef Documentation

### 6.36.1.1 `tensor_type`

```
template<Tensor Tsor>
typedef Tsor ceras::variable< Tsor >::tensor_type
```

### 6.36.1.2 `value_type`

```
template<Tensor Tsor>
typedef tensor_type::value_type ceras::variable< Tsor >::value_type
```

## 6.36.2 Constructor & Destructor Documentation

**6.36.2.1 variable()** [1/4]

```
template<Tensor Tzor>
ceras::variable< Tzor >::variable (
    tensor_type const & data,
    value_type l1 = value_type{0},
    value_type l2 = value_type{0},
    bool trainable = true ) [inline]
```

**6.36.2.2 variable()** [2/4]

```
template<Tensor Tzor>
ceras::variable< Tzor >::variable ( ) [delete]
```

**6.36.2.3 variable()** [3/4]

```
template<Tensor Tzor>
ceras::variable< Tzor >::variable (
    variable< Tzor > const & other ) [default]
```

**6.36.2.4 variable()** [4/4]

```
template<Tensor Tzor>
ceras::variable< Tzor >::variable (
    variable< Tzor > && ) [default]
```

**6.36.3 Member Function Documentation****6.36.3.1 backward()**

```
template<Tensor Tzor>
void ceras::variable< Tzor >::backward (
    auto const & grad ) [inline], [noexcept]
```

**6.36.3.2 contexts()** [1/2]

```
template<Tensor Tzor>
std::vector<tensor_type>& ceras::variable< Tzor >::contexts ( ) [inline]
```

### 6.36.3.3 contexts() [2/2]

```
template<Tensor Tsor>
std::vector<tensor_type> ceras::variable< Tsor >::contexts ( ) const [inline]
```

### 6.36.3.4 data() [1/2]

```
template<Tensor Tsor>
tensor_type& ceras::variable< Tsor >::data ( ) [inline]
```

### 6.36.3.5 data() [2/2]

```
template<Tensor Tsor>
tensor_type ceras::variable< Tsor >::data ( ) const [inline]
```

### 6.36.3.6 forward()

```
template<Tensor Tsor>
tensor_type const ceras::variable< Tsor >::forward ( ) [inline], [noexcept]
```

### 6.36.3.7 gradient() [1/2]

```
template<Tensor Tsor>
tensor_type& ceras::variable< Tsor >::gradient ( ) [inline]
```

### 6.36.3.8 gradient() [2/2]

```
template<Tensor Tsor>
tensor_type ceras::variable< Tsor >::gradient ( ) const [inline]
```

### 6.36.3.9 operator=() [1/2]

```
template<Tensor Tsor>
variable& ceras::variable< Tsor >::operator= (
    variable< Tsor > && ) [default]
```



#### 6.36.3.10 operator=() [2/2]

```
template<Tensor Tzor>
variable& ceras::variable< Tzor >::operator= (
    variable< Tzor > const & other ) [default]
```

#### 6.36.3.11 reset()

```
template<Tensor Tzor>
void ceras::variable< Tzor >::reset ( ) [inline]
```

#### 6.36.3.12 shape()

```
template<Tensor Tzor>
std::vector<std::size_t> ceras::variable< Tzor >::shape ( ) const [inline], [noexcept]
```

#### 6.36.3.13 trainable() [1/2]

```
template<Tensor Tzor>
bool ceras::variable< Tzor >::trainable ( ) const [inline], [noexcept]
```

#### 6.36.3.14 trainable() [2/2]

```
template<Tensor Tzor>
void ceras::variable< Tzor >::trainable (
    bool t ) [inline]
```

### 6.36.4 Member Data Documentation

#### 6.36.4.1 regularizer\_

```
template<Tensor Tzor>
regularizer<value_type> ceras::variable< Tzor >::regularizer_
```

#### 6.36.4.2 state\_

```
template<Tensor Tsor>
std::shared_ptr<variable_state<tensor_type> > ceras::variable< Tsor >::state_
```

#### 6.36.4.3 trainable\_

```
template<Tensor Tsor>
bool ceras::variable< Tsor >::trainable_
```

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

- /home/feng/workspace/github.repo/ceras/include/[variable.hpp](#)

### 6.37 ceras::variable\_state< Tsor > Struct Template Reference

```
#include <variable.hpp>
```

#### Public Attributes

- Tsor [data\\_](#)
- Tsor [gradient\\_](#)
- std::vector< Tsor > [contexts\\_](#)

#### 6.37.1 Member Data Documentation

##### 6.37.1.1 contexts\_

```
template<Tensor Tsor>
std::vector<Tsor> ceras::variable_state< Tsor >::contexts_
```

##### 6.37.1.2 data\_

```
template<Tensor Tsor>
Tsor ceras::variable_state< Tsor >::data_
```

### 6.37.1.3 gradient\_

```
template<Tensor Tsor>
Tsor ceras::variable_state< Tsor >::gradient_
```

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

- </home/feng/workspace/github.repo/ceras/include/variable.hpp>

## 6.38 ceras::view\_2d< T > Struct Template Reference

```
#include <tensor.hpp>
```

### Public Types

- typedef T [value\\_type](#)
- typedef [value\\_type](#) \* [row\\_type](#)
- typedef const [value\\_type](#) \* [const\\_row\\_type](#)
- typedef stride\_iterator< [value\\_type](#) \* > [col\\_type](#)
- typedef stride\_iterator< const [value\\_type](#) \* > [const\\_col\\_type](#)

### Public Member Functions

- template<typename A >  
constexpr [view\\_2d](#) ([tensor](#)< T, A > &tsor, unsigned long [row](#), unsigned long [col](#), bool transposed=false) noexcept
- constexpr [view\\_2d](#) (T \*[data](#), unsigned long [row](#), unsigned long [col](#), bool transposed=false) noexcept
- constexpr [view\\_2d](#) (const T \*[data](#), unsigned long [row](#), unsigned long [col](#), bool transposed=false) noexcept
- constexpr T \* [operator\[\]](#) (unsigned long index)
- constexpr const T \* [operator\[\]](#) (unsigned long index) const
- constexpr auto [shape](#) () const noexcept
- constexpr unsigned long [size](#) () const noexcept
- constexpr T \* [data](#) () noexcept
- constexpr const T \* [data](#) () const noexcept
- constexpr T \* [begin](#) () noexcept
- constexpr const T \* [end](#) () const noexcept
- constexpr unsigned long [row](#) () const noexcept
- constexpr unsigned long [col](#) () const noexcept
- constexpr [row\\_type](#) [row\\_begin](#) (unsigned long index=0) noexcept
- constexpr [row\\_type](#) [row\\_end](#) (unsigned long index=0) noexcept
- constexpr [const\\_row\\_type](#) [row\\_begin](#) (unsigned long index=0) const noexcept
- constexpr [const\\_row\\_type](#) [row\\_end](#) (unsigned long index=0) const noexcept
- constexpr [col\\_type](#) [col\\_begin](#) (unsigned long index=0) noexcept
- constexpr [col\\_type](#) [col\\_end](#) (unsigned long index=0) noexcept
- constexpr [const\\_col\\_type](#) [col\\_begin](#) (unsigned long index=0) const noexcept
- constexpr [const\\_col\\_type](#) [col\\_end](#) (unsigned long index=0) const noexcept

## Public Attributes

- T \* [data\\_](#)
- unsigned long [row\\_](#)
- unsigned long [col\\_](#)
- bool [transposed\\_](#)

## 6.38.1 Member Typedef Documentation

### 6.38.1.1 col\_type

```
template<typename T >  
typedef stride_iterator<value_type*> ceras::view_2d< T >::col_type
```

### 6.38.1.2 const\_col\_type

```
template<typename T >  
typedef stride_iterator<const value_type*> ceras::view_2d< T >::const_col_type
```

### 6.38.1.3 const\_row\_type

```
template<typename T >  
typedef const value_type* ceras::view_2d< T >::const_row_type
```

### 6.38.1.4 row\_type

```
template<typename T >  
typedef value_type* ceras::view_2d< T >::row_type
```

### 6.38.1.5 value\_type

```
template<typename T >  
typedef T ceras::view_2d< T >::value_type
```

## 6.38.2 Constructor & Destructor Documentation

**6.38.2.1 view\_2d()** [1/3]

```
template<typename T >
template<typename A >
constexpr ceras::view_2d< T >::view_2d (
    tensor< T, A > & tsor,
    unsigned long row,
    unsigned long col,
    bool transposed = false ) [inline], [constexpr], [noexcept]
```

**6.38.2.2 view\_2d()** [2/3]

```
template<typename T >
constexpr ceras::view_2d< T >::view_2d (
    T * data,
    unsigned long row,
    unsigned long col,
    bool transposed = false ) [inline], [constexpr], [noexcept]
```

**6.38.2.3 view\_2d()** [3/3]

```
template<typename T >
constexpr ceras::view_2d< T >::view_2d (
    const T * data,
    unsigned long row,
    unsigned long col,
    bool transposed = false ) [inline], [constexpr], [noexcept]
```

**6.38.3 Member Function Documentation****6.38.3.1 begin()**

```
template<typename T >
constexpr T* ceras::view_2d< T >::begin ( ) [inline], [constexpr], [noexcept]
```

**6.38.3.2 col()**

```
template<typename T >
constexpr unsigned long ceras::view_2d< T >::col ( ) const [inline], [constexpr], [noexcept]
```

**6.38.3.3 col\_begin() [1/2]**

```
template<typename T >
constexpr const\_col\_type ceras::view\_2d< T >::col_begin (
    unsigned long index = 0 ) const [inline], [constexpr], [noexcept]
```

**6.38.3.4 col\_begin() [2/2]**

```
template<typename T >
constexpr col\_type ceras::view\_2d< T >::col_begin (
    unsigned long index = 0 ) [inline], [constexpr], [noexcept]
```

**6.38.3.5 col\_end() [1/2]**

```
template<typename T >
constexpr const\_col\_type ceras::view\_2d< T >::col_end (
    unsigned long index = 0 ) const [inline], [constexpr], [noexcept]
```

**6.38.3.6 col\_end() [2/2]**

```
template<typename T >
constexpr col\_type ceras::view\_2d< T >::col_end (
    unsigned long index = 0 ) [inline], [constexpr], [noexcept]
```

**6.38.3.7 data() [1/2]**

```
template<typename T >
constexpr const T* ceras::view\_2d< T >::data ( ) const [inline], [constexpr], [noexcept]
```

**6.38.3.8 data() [2/2]**

```
template<typename T >
constexpr T* ceras::view\_2d< T >::data ( ) [inline], [constexpr], [noexcept]
```

### 6.38.3.9 end()

```
template<typename T >
constexpr const T* ceras::view_2d< T >::end ( ) const [inline], [constexpr], [noexcept]
```

### 6.38.3.10 operator[]() [1/2]

```
template<typename T >
constexpr T* ceras::view_2d< T >::operator[] (
    unsigned long index ) [inline], [constexpr]
```

### 6.38.3.11 operator[]() [2/2]

```
template<typename T >
constexpr const T* ceras::view_2d< T >::operator[] (
    unsigned long index ) const [inline], [constexpr]
```

### 6.38.3.12 row()

```
template<typename T >
constexpr unsigned long ceras::view_2d< T >::row ( ) const [inline], [constexpr], [noexcept]
```

### 6.38.3.13 row\_begin() [1/2]

```
template<typename T >
constexpr const_row_type ceras::view_2d< T >::row_begin (
    unsigned long index = 0 ) const [inline], [constexpr], [noexcept]
```

### 6.38.3.14 row\_begin() [2/2]

```
template<typename T >
constexpr row_type ceras::view_2d< T >::row_begin (
    unsigned long index = 0 ) [inline], [constexpr], [noexcept]
```

#### 6.38.3.15 row\_end() [1/2]

```
template<typename T >
constexpr const_row_type ceras::view_2d< T >::row_end (
    unsigned long index = 0 ) const [inline], [constexpr], [noexcept]
```

#### 6.38.3.16 row\_end() [2/2]

```
template<typename T >
constexpr row_type ceras::view_2d< T >::row_end (
    unsigned long index = 0 ) [inline], [constexpr], [noexcept]
```

#### 6.38.3.17 shape()

```
template<typename T >
constexpr auto ceras::view_2d< T >::shape ( ) const [inline], [constexpr], [noexcept]
```

#### 6.38.3.18 size()

```
template<typename T >
constexpr unsigned long ceras::view_2d< T >::size ( ) const [inline], [constexpr], [noexcept]
```

### 6.38.4 Member Data Documentation

#### 6.38.4.1 col\_

```
template<typename T >
unsigned long ceras::view_2d< T >::col_
```

#### 6.38.4.2 data\_

```
template<typename T >
T* ceras::view_2d< T >::data_
```



#### 6.38.4.3 row\_

```
template<typename T >
unsigned long ceras::view_2d< T >::row_
```

#### 6.38.4.4 transposed\_

```
template<typename T >
bool ceras::view_2d< T >::transposed_
```

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

- </home/feng/workspace/github.repo/ceras/include/tensor.hpp>

## 6.39 ceras::view\_3d< T > Struct Template Reference

```
#include <tensor.hpp>
```

### Public Member Functions

- constexpr [view\\_3d](#) (T \*data, unsigned long row, unsigned long col, unsigned long channel) noexcept
- constexpr auto [operator\[\]](#) (unsigned long index) noexcept
- constexpr auto [operator\[\]](#) (unsigned long index) const noexcept

### Public Attributes

- T \* [data\\_](#)
- unsigned long [row\\_](#)
- unsigned long [col\\_](#)
- unsigned long [channel\\_](#)

### 6.39.1 Constructor & Destructor Documentation

#### 6.39.1.1 view\_3d()

```
template<typename T >
constexpr ceras::view_3d< T >::view_3d (
    T * data,
    unsigned long row,
    unsigned long col,
    unsigned long channel ) [inline], [constexpr], [noexcept]
```

## 6.39.2 Member Function Documentation

### 6.39.2.1 operator[]() [1/2]

```
template<typename T >
constexpr auto ceras::view_3d< T >::operator[] (
    unsigned long index ) const [inline], [constexpr], [noexcept]
```

### 6.39.2.2 operator[]() [2/2]

```
template<typename T >
constexpr auto ceras::view_3d< T >::operator[] (
    unsigned long index ) [inline], [constexpr], [noexcept]
```

## 6.39.3 Member Data Documentation

### 6.39.3.1 channel\_

```
template<typename T >
unsigned long ceras::view_3d< T >::channel_
```

### 6.39.3.2 col\_

```
template<typename T >
unsigned long ceras::view_3d< T >::col_
```

### 6.39.3.3 data\_

```
template<typename T >
T* ceras::view_3d< T >::data_
```

### 6.39.3.4 row\_

```
template<typename T >
unsigned long ceras::view_3d< T >::row_
```

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

- /home/feng/workspace/github.repo/ceras/include/[tensor.hpp](#)

## 6.40 ceras::view\_4d< T > Struct Template Reference

```
#include <tensor.hpp>
```

### Public Member Functions

- constexpr [view\\_4d](#) (T \*data, unsigned long batch\_size, unsigned long row, unsigned long col, unsigned long channel) noexcept
- constexpr auto [operator\[\]](#) (unsigned long index) noexcept
- constexpr auto [operator\[\]](#) (unsigned long index) const noexcept

### Public Attributes

- T \* [data\\_](#)  
*The pointer to the start position of the 1-D array.*
- unsigned long [batch\\_size\\_](#)  
*The batch size of the 4-D tensor, also the first dimension of the tensor.*
- unsigned long [row\\_](#)  
*The row of the 4-D tensor, also the second dimension of the tensor.*
- unsigned long [col\\_](#)  
*The column of the 4-D tensor, also the third dimension of the tensor.*
- unsigned long [channel\\_](#)  
*The channel of the 4-D tensor, also the last dimension of the tensor.*

### 6.40.1 Detailed Description

```
template<typename T>
struct ceras::view_4d< T >
```

A class viewing a 1-D array as a 4-D tensor. This class is useful when treating an array as a typical 4-D tensor in a neural network, with a shape of [batch\_size, row, column, channel].

### 6.40.2 Constructor & Destructor Documentation

#### 6.40.2.1 `view_4d()`

```
template<typename T >
constexpr ceras::view\_4d< T >::view_4d (
    T * data,
    unsigned long batch_size,
    unsigned long row,
    unsigned long col,
    unsigned long channel ) [inline], [constexpr], [noexcept]
```

Constructor of [view\\_4d](#)

## Parameters

<i>data</i>	The raw pointer to the start position of the 1-D array.
<i>batch_size</i>	The first dimension of the 4-D tensor, also for the batch size in the CNN layers.
<i>row</i>	The second dimension of the 4-D tensor, also for the row in the CNN layers.
<i>col</i>	The third dimension of the 4-D tensor, also for the column in the CNN layers.
<i>channel</i>	The last dimension of the 4-D tensor, also for the channel in the CNN layers.

### 6.40.3 Member Function Documentation

#### 6.40.3.1 operator[]() [1/2]

```
template<typename T >
constexpr auto ceras::view_4d< T >::operator[] (
    unsigned long index ) const [inline], [constexpr], [noexcept]
```

Giving a [view\\_3d](#) interface for operator [].

## Parameters

<i>index</i>	The first dimension of the 4-D tensor.
--------------	--

## Example usage:

```
std::vector<float> array;
array.resize( 16*8*8*3 );
// operations on 'array'
auto t = view_4d{ array.data(), 16, 8, 8, 3 };
float v0123 = t[0][1][2][3];
```

#### 6.40.3.2 operator[]() [2/2]

```
template<typename T >
constexpr auto ceras::view_4d< T >::operator[] (
    unsigned long index ) [inline], [constexpr], [noexcept]
```

Giving a [view\\_3d](#) interface for operator [].

## Parameters

<i>index</i>	The first dimension of the 4-D tensor.
--------------	--

## Example usage:

```
std::vector<float> array;
array.resize( 16*8*8*3 );
auto t = view_4d{ array.data(), 16, 8, 8, 3 };
t[0][1][2][3] = 1.0;
```

## 6.40.4 Member Data Documentation

### 6.40.4.1 batch\_size\_

```
template<typename T >
unsigned long ceras::view_4d< T >::batch_size_
```

The batch size of the 4-D tensor, also the first dimension of the tensor.

### 6.40.4.2 channel\_

```
template<typename T >
unsigned long ceras::view_4d< T >::channel_
```

The channel of the 4-D tensor, also the last dimension of the tensor.

### 6.40.4.3 col\_

```
template<typename T >
unsigned long ceras::view_4d< T >::col_
```

The column of the 4-D tensor, also the third dimension of the tensor.

### 6.40.4.4 data\_

```
template<typename T >
T* ceras::view_4d< T >::data_
```

The pointer to the start position of the 1-D array.

### 6.40.4.5 row\_

```
template<typename T >
unsigned long ceras::view_4d< T >::row_
```

The row of the 4-D tensor, also the second dimension of the tensor.

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

- </home/feng/workspace/github.repo/ceras/include/tensor.hpp>

## Chapter 7

# File Documentation

### 7.1 /home/feng/workspace/github.repo/ceras/include/activation.hpp File Reference

```
#include "../operation.hpp"
#include "../tensor.hpp"
#include "../utils/range.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/for_each.hpp"
#include "../utils/context_cast.hpp"
```

#### Namespaces

- [ceras](#)

#### Functions

- `template<Expression Ex>`  
`constexpr auto ceras::softmax (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`auto ceras::selu (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`auto ceras::softplus (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`auto ceras::softsign (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`auto ceras::sigmoid (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`auto ceras::relu (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`auto ceras::relu6 (Ex const &ex) noexcept`
- `template<typename T >`  
`requires std::floating_point< T > auto ceras::leaky\_relu (T const factor) noexcept`
- `template<Expression Ex>`  
`auto ceras::negative\_relu (Ex const &ex) noexcept`

- `template<typename T = float>`  
`requires std::floating_point< T >` `auto ceras::elu` (T const alpha=1.0) noexcept
- `template<Expression Ex>`  
`auto ceras::exponential` (Ex const &ex) noexcept
- `template<Expression Ex>`  
`auto ceras::hard_sigmoid` (Ex const &ex) noexcept
- `template<Expression Ex>`  
`auto ceras::gelu` (Ex const &ex) noexcept
- `template<Expression Ex>`  
`auto ceras::swish` (Ex const &ex) noexcept  
*Applies the swish activation function.*
- `template<Expression Ex>`  
`auto ceras::silu` (Ex const &ex) noexcept  
*An alias name of activation swish.*
- `template<Expression Ex>`  
`auto ceras::crelu` (Ex const &ex) noexcept  
*Concatenated Rectified Linear Units, an activation function which preserves both positive and negative phase information while enforcing non-saturated non-linearity.*

## 7.2 /home/feng/workspace/github.repo/ceras/include/ceras.hpp File Reference

```
#include "../config.hpp"
#include "../includes.hpp"
#include "../activation.hpp"
#include "../ceras.hpp"
#include "../loss.hpp"
#include "../operation.hpp"
#include "../complex_operator.hpp"
#include "../optimizer.hpp"
#include "../place_holder.hpp"
#include "../session.hpp"
#include "../tensor.hpp"
#include "../variable.hpp"
#include "../constant.hpp"
#include "../layer.hpp"
#include "../model.hpp"
#include "../dataset.hpp"
```

## 7.3 /home/feng/workspace/github.repo/ceras/include/complex\_↵operator.hpp File Reference

```
#include "../operation.hpp"
```

### Classes

- struct `ceras::complex< Real_Ex, Imag_Ex >`
- struct `ceras::is_complex< T >`
- struct `ceras::is_complex< complex< Real_Ex, Imag_Ex > >`



## Namespaces

- [ceras](#)

## Functions

- `template<Expression Real_Ex, Expression Imag_Ex>`  
`Real_Ex ceras::real (complex< Real_Ex, Imag_Ex > const &c) noexcept`
- `template<Expression Real_Ex, Expression Imag_Ex>`  
`Imag_Ex ceras::imag (complex< Real_Ex, Imag_Ex > const &c) noexcept`
- `template<Complex C>`  
`auto ceras::abs (C const &c) noexcept`  
*Returns the magnitude of the complex expression.*
- `template<Complex C>`  
`auto ceras::norm (C const &c) noexcept`  
*Returns the squared magnitude of the complex expression.*
- `template<Complex C>`  
`auto ceras::conj (C const &c) noexcept`  
*Returns the conjugate of the complex expression.*
- `template<Expression Em, Expression Ep>`  
`auto ceras::polar (Em const &em, Ep const &ep) noexcept`  
*Returns with given magnitude and phase angle.*
- `template<Complex C>`  
`auto ceras::arg (C const &c) noexcept`  
*Calculates the phase angle (in radians) of the complex expression.*
- `template<Complex C>`  
`auto ceras::operator+ (C const &c) noexcept`  
*Returns the complex expression.*
- `template<Complex C>`  
`auto ceras::operator- (C const &c) noexcept`  
*Negatives the complex expression.*
- `template<Complex Cl, Complex Cr>`  
`auto ceras::operator+ (Cl const &cl, Cr const &cr) noexcept`  
*Sums up two complex expressions.*
- `template<Complex Cl, Complex Cr>`  
`auto ceras::operator- (Cl const &cl, Cr const &cr) noexcept`  
*Subtracts one complex expression from the other one.*
- `template<Complex Cl, Complex Cr>`  
`auto ceras::operator\* (Cl const &cl, Cr const &cr) noexcept`  
*Multiplies two complex expressions. Optimization here:  $(a+ib)*(c+id) = (ac-bd) + i(ad+bc) = (ac-bd) + i((a+b)*(c+d)-ac-bd)$*
- `template<Complex C, Expression E>`  
`auto ceras::operator+ (C const &c, E const &e) noexcept`  
*Sums up a complex expression and an expression.*
- `template<Complex C, Expression E>`  
`auto ceras::operator+ (E const &e, C const &c) noexcept`  
*Sums up a complex expression and an expression.*
- `template<Complex C, Expression E>`  
`auto ceras::operator- (C const &c, E const &e) noexcept`  
*Subtracts an expression from a complex expression.*
- `template<Complex C, Expression E>`  
`auto ceras::operator- (E const &e, C const &c) noexcept`  
*Subtracts a complex expression from an expression.*

- `template<Complex C, Expression E>`  
`auto ceras::operator\* (C const &c, E const &e) noexcept`  
*Multiplies a complex expression with an expression.*
- `template<Complex C, Expression E>`  
`auto ceras::operator\* (E const &e, C const &c) noexcept`  
*Multiplies an expression with a compression expression.*

## Variables

- `template<typename T >`  
`constexpr bool ceras::is\_complex\_v = is_complex<T>::value`
- `template<typename T >`  
`concept ceras::Complex = is_complex_v<T>`  
*A type that represents a complex expression.*

## 7.4 /home/feng/workspace/github.repo/ceras/include/config.hpp File Reference

## 7.5 /home/feng/workspace/github.repo/ceras/include/constant.hpp File Reference

```
#include "../includes.hpp"
#include "../tensor.hpp"
#include "../utils/id.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/enable_shared.hpp"
```

## Classes

- `struct ceras::constant< Tsor >`  
*Creates a constant expression from a tensor-like object.*
- `struct ceras::is\_constant< T >`
- `struct ceras::is\_constant< constant< Tsor > >`

## Namespaces

- [ceras](#)

## Variables

- `template<class T >`  
`constexpr bool ceras::is\_constant\_v = is_constant<T>::value`
- `template<typename T >`  
`concept ceras::Constant = is_constant_v<T>`

## 7.6 /home/feng/workspace/github.repo/ceras/include/dataset.hpp File Reference

```
#include "../tensor.hpp"
#include "../includes.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/for_each.hpp"
```

### Namespaces

- [ceras](#)
- [ceras::dataset](#)
- [ceras::dataset::mnist](#)
- [ceras::dataset::fashion\\_mnist](#)

### Functions

- auto [ceras::dataset::mnist::load\\_data](#) (std::string const &path=std::string{"./dataset/mnist"})
- auto [ceras::dataset::fashion\\_mnist::load\\_data](#) (std::string const &path=std::string{"./dataset/fashion\_mnist"})

## 7.7 /home/feng/workspace/github.repo/ceras/include/includes.hpp File Reference

```
#include "../config.hpp"
#include <algorithm>
#include <any>
#include <array>
#include <cassert>
#include <chrono>
#include <cmath>
#include <compare>
#include <concepts>
#include <cstdint>
#include <ctime>
#include <filesystem>
#include <fstream>
#include <functional>
#include <initializer_list>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <limits>
#include <map>
#include <memory>
#include <numeric>
#include <optional>
#include <ostream>
#include <random>
#include <ranges>
#include <regex>
```

```
#include <set>
#include <sstream>
#include <string>
#include <tuple>
#include <thread>
#include <type_traits>
#include <unordered_map>
#include <unordered_set>
#include <utility>
#include <vector>
#include "../utils/3rd_party/stb_image.h"
#include "../utils/3rd_party/stb_image_write.h"
#include "../utils/3rd_party/stb_image_resize.h"
#include "../utils/3rd_party/glob.hpp"
```

## Macros

- `#define STB_IMAGE_IMPLEMENTATION`
- `#define STB_IMAGE_WRITE_IMPLEMENTATION`
- `#define STB_IMAGE_RESIZE_IMPLEMENTATION`

### 7.7.1 Macro Definition Documentation

#### 7.7.1.1 STB\_IMAGE\_IMPLEMENTATION

```
#define STB_IMAGE_IMPLEMENTATION
```

#### 7.7.1.2 STB\_IMAGE\_RESIZE\_IMPLEMENTATION

```
#define STB_IMAGE_RESIZE_IMPLEMENTATION
```

#### 7.7.1.3 STB\_IMAGE\_WRITE\_IMPLEMENTATION

```
#define STB_IMAGE_WRITE_IMPLEMENTATION
```

## 7.8 /home/feng/workspace/github.repo/ceras/include/layer.hpp File Reference

```
#include "../operation.hpp"
#include "../activation.hpp"
#include "../loss.hpp"
#include "../optimizer.hpp"
#include "../utils/better_assert.hpp"
```

## Namespaces

- [ceras](#)

## Functions

- auto [ceras::Input](#) ()
- auto [ceras::Conv2D](#) (unsigned long output\_channels, std::vector< unsigned long > const &kernel\_size, std::vector< unsigned long > const &input\_shape, std::string const &padding="valid", std::vector< unsigned long > const &strides={1, 1}, std::vector< unsigned long > const &dilations={1, 1}, bool use\_bias=true, float kernel\_regularizer\_l1=0.0f, float kernel\_regularizer\_l2=0.0f, float bias\_regularizer\_l1=0.0f, float bias\_regularizer\_l2=0.0f)
 

*2D convolution layer.*
- auto [ceras::Dense](#) (unsigned long output\_size, unsigned long input\_size, bool use\_bias=true, float kernel\_regularizer\_l1=0.0f, float kernel\_regularizer\_l2=0.0f, float bias\_regularizer\_l1=0.0f, float bias\_regularizer\_l2=0.0f)
 

*Densely-connected layer.*
- auto [ceras::BatchNormalization](#) (std::vector< unsigned long > const &shape, float threshold=0.95f, float kernel\_regularizer\_l1=0.0f, float kernel\_regularizer\_l2=0.0f, float bias\_regularizer\_l1=0.0f, float bias\_regularizer\_l2=0.0f)
 

*Applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.*
- auto [ceras::BatchNormalization](#) (float threshold, std::vector< unsigned long > const &shape, float kernel\_regularizer\_l1=0.0f, float kernel\_regularizer\_l2=0.0f, float bias\_regularizer\_l1=0.0f, float bias\_regularizer\_l2=0.0f)
- auto [ceras::Concatenate](#) (unsigned long axis=-1) noexcept
- auto [ceras::Add](#) () noexcept
- auto [ceras::Subtract](#) () noexcept
- auto [ceras::Multiply](#) () noexcept
- template<Expression Ex>
 

auto [ceras::ReLU](#) (Ex const &ex) noexcept
- auto [ceras::Softmax](#) () noexcept
- template<typename T = float>
 

auto [ceras::LeakyReLU](#) (T const factor=0.2) noexcept
- template<typename T = float>
 

auto [ceras::ELU](#) (T const factor=0.2) noexcept
- auto [ceras::Reshape](#) (std::vector< unsigned long > const &new\_shape, bool include\_batch\_flag=true) noexcept
- auto [ceras::Flatten](#) () noexcept
- auto [ceras::MaxPooling2D](#) (unsigned long stride) noexcept
- auto [ceras::UpSampling2D](#) (unsigned long stride) noexcept
- template<typename T >
 

auto [ceras::Dropout](#) (T factor) noexcept
- auto [ceras::AveragePooling2D](#) (unsigned long stride) noexcept

## 7.9 /home/feng/workspace/github.repo/ceras/include/loss.hpp File Reference

```
#include "../operation.hpp"
#include "../tensor.hpp"
#include "../utils/debug.hpp"
```

## Namespaces

- [ceras](#)

## Functions

- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::mean\_squared\_logarithmic\_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::squared\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::mean\_squared\_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::mse (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::abs\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::mean\_absolute\_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::mae (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::cross\_entropy (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::binary\_cross\_entropy\_loss (Lhs_Expression const &ground_truth, Rhs_Expression const &prediction) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::cross\_entropy\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::hinge\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`

## Variables

- `auto ceras::MeanSquaredError`  
*Computes the mean of squares of errors between labels and predictions.*
- `auto ceras::MSE`  
*An alias name of function [MeanSquaredError](#).*
- `auto ceras::MeanAbsoluteError`  
*Computes the mean of absolute errors between labels and predictions.*
- `auto ceras::MAE`  
*An alias name of function [MeanAbsoluteError](#).*
- `auto ceras::Hinge`
- `auto ceras::CategoricalCrossentropy`
- `auto ceras::CategoricalCrossEntropy`
- `auto ceras::BinaryCrossentropy`
- `auto ceras::BinaryCrossEntropy`

## 7.10 /home/feng/workspace/github.repo/ceras/include/model.hpp File Reference

```
#include "../includes.hpp"
#include "../operation.hpp"
#include "../place_holder.hpp"
#include "../tensor.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/context_cast.hpp"
#include "../utils/tqdm.hpp"
```

### Classes

- struct [ceras::compiled\\_model](#)< [Model](#), [Optimizer](#), [Loss](#) >
- struct [ceras::model](#)< [Ex](#), [Ph](#) >

### Namespaces

- [ceras](#)

### Functions

- template<Expression [Ex](#)>  
void [ceras::make\\_trainable](#) ([Ex](#) &[ex](#), bool [t](#))
- template<Expression [Ex](#), [Place\\_Holder](#) [Ph](#), Expression [Ey](#)>  
auto [ceras::replace\\_placeholder\\_with\\_expression](#) ([Ex](#) const &[ex](#), [Ph](#) const &[old\\_place\\_holder](#), [Ey](#) const &[new\\_expression](#))
- template<typename [Model](#) , typename [Optimizer](#) , typename [Loss](#) >  
auto [ceras::make\\_compiled\\_model](#) ([Model](#) const &[m](#), [Loss](#) const &[l](#), [Optimizer](#) const &[o](#))

## 7.11 /home/feng/workspace/github.repo/ceras/include/operation.hpp File Reference

```
#include "../includes.hpp"
#include "../place_holder.hpp"
#include "../variable.hpp"
#include "../constant.hpp"
#include "../value.hpp"
#include "../utils/range.hpp"
#include "../utils/debug.hpp"
#include "../config.hpp"
#include "../utils/context_cast.hpp"
#include "../utils/for_each.hpp"
#include "../utils/id.hpp"
#include "../utils/enable_shared.hpp"
```

## Classes

- struct [ceras::unary\\_operator](#)< Operator, Forward\_Action, Backward\_Action >
- struct [ceras::binary\\_operator](#)< Lhs\_Operator, Rhc\_Operator, Forward\_Action, Backward\_Action >
- struct [ceras::is\\_unary\\_operator](#)< T >
- struct [ceras::is\\_unary\\_operator](#)< unary\_operator< Operator, Forward\_Action, Backward\_Action > >
- struct [ceras::is\\_binary\\_operator](#)< T >
- struct [ceras::is\\_binary\\_operator](#)< binary\_operator< Lhs\_Operator, Rhc\_Operator, Forward\_Action, Backward\_Action > >

## Namespaces

- [ceras](#)

## Functions

- template<Expression Ex>  
std::string [ceras::computation\\_graph](#) (Ex const &ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::plus](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::operator+](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::operator+](#) (Ex const &ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
auto [ceras::operator\\*](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::negative](#) (Ex const &ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::operator-](#) (Ex const &ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::elementwise\\_product](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::elementwise\\_multiply](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::hadamard\\_product](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::sum\\_reduce](#) (Ex const &ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::reduce\\_sum](#) (Ex const &ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::mean\\_reduce](#) (Ex const &ex) noexcept  
*Computes the mean of elements across all dimensions of an expression.*
- template<Expression Ex>  
constexpr auto [ceras::reduce\\_mean](#) (Ex const &ex) noexcept  
*An alias name of mean\_reduce.*
- template<Expression Ex>  
constexpr auto [ceras::mean](#) (Ex const &ex) noexcept  
*An alias name of mean\_reduce.*
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::minus](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex) noexcept



- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto ceras::operator- (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto ceras::square (Ex const &ex) noexcept`
- `template<Expression Ex, Expression Ey>`  
`*endcode **constexpr auto hypot (Ex const &ex, Ey const &ey) noexcept`
- `template<typename Float >`  
`requires std::floating_point< Float > constexpr auto clip (Float lower, Float upper=std::numeric_limits< Float >::max()) noexcept`
- `auto reshape (std::vector< unsigned long > const &new_shape, bool include_batch_flag=true) noexcept`
- `template<Expression Ex>`  
`constexpr auto flatten (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto identity (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`auto transpose (Ex const &ex) noexcept`
- `auto img2col (unsigned long const row_kernel, unsigned long col_kernel=-1, unsigned long const row_stride=1, unsigned long const col_stride=1, unsigned long const row_dilation=1, unsigned long const col_dilation=1) noexcept`
- `auto conv2d (unsigned long row_input, unsigned long col_input, unsigned long const row_stride=1, unsigned long const col_stride=1, unsigned long const row_dilation=1, unsigned long const col_dilation=1, std::string const &padding="valid") noexcept`
- `template<typename T >`  
`requires std::floating_point< T > auto drop\_out (T const factor) noexcept`
- `auto max\_pooling\_2d (unsigned long stride) noexcept`
- `auto average\_pooling\_2d (unsigned long stride) noexcept`
- `auto up\_sampling\_2d (unsigned long stride) noexcept`
- `template<typename T = double>`  
`requires std::floating_point< T > auto normalization\_batch (T const momentum=0.98) noexcept`
- `template<typename T >`  
`requires std::floating_point< T > auto batch\_normalization (T const momentum=0.98) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto concatenate (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `auto concatenate (unsigned long axe=-1)`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto concat (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `auto concat (unsigned long axe=-1)`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto maximum (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto atan2 (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`  
*Computes the arc tangent of y/x using the signs of arguments to determine the correct quadrant.*
- `template<typename T = float>`  
`requires std::floating_point< T > auto random\_normal\_like (T mean=0.0, T stddev=1.0) noexcept`
- `template<Expression Ex>`  
`auto ones\_like (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`auto zeros\_like (Ex const &ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression, std::floating_point FP>`  
`constexpr auto equal (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex, FP threshold=0.5) noexcept`
- `template<Expression Ex>`  
`constexpr auto sign (Ex const &ex) noexcept`
- `auto zero\_padding\_2d (std::vector< unsigned long > const &padding) noexcept`  
*Zero-padding layer for 2D input. The input should have 4-dimensions: (batch\_size, row, col, channel). The output has 4-dimensions: (batch\_size, new\_row, new\_col, channel).*

- auto [repeat](#) (unsigned long repeats, unsigned long axis=-1) noexcept  
*Repeats elements along an axis.*
- auto [reduce\\_min](#) (unsigned long axis=-1) noexcept  
*Reduce minimal elements along an axis.*
- auto [reduce\\_max](#) (unsigned long axis=-1) noexcept  
*Reduce maximum elements along an axis.*
- auto [reduce\\_sum](#) (unsigned long axis) noexcept  
*Reduce sum elements along an axis.*
- template<Expression Ex>  
constexpr auto [abs](#) (Ex const &ex) noexcept  
*Computes Abs of the given expression.*
- template<Expression Ex>  
constexpr auto [acos](#) (Ex const &ex) noexcept  
*Computes Acos of the given expression.*
- template<Expression Ex>  
constexpr auto [acosh](#) (Ex const &ex) noexcept  
*Computes Acosh of the given expression.*
- template<Expression Ex>  
constexpr auto [asin](#) (Ex const &ex) noexcept  
*Computes Asin of the given expression.*
- template<Expression Ex>  
constexpr auto [asinh](#) (Ex const &ex) noexcept  
*Computes Asinh of the given expression.*
- template<Expression Ex>  
constexpr auto [atan](#) (Ex const &ex) noexcept  
*Computes Atan of the given expression.*
- template<Expression Ex>  
constexpr auto [atanh](#) (Ex const &ex) noexcept  
*Computes Atanh of the given expression.*
- template<Expression Ex>  
constexpr auto [cbrt](#) (Ex const &ex) noexcept  
*Computes Cbert of the given expression.*
- template<Expression Ex>  
constexpr auto [ceil](#) (Ex const &ex) noexcept  
*Computes Ceil of the given expression.*
- template<Expression Ex>  
constexpr auto [cos](#) (Ex const &ex) noexcept  
*Computes Cos of the given expression.*
- template<Expression Ex>  
constexpr auto [cosh](#) (Ex const &ex) noexcept  
*Computes Cosh of the given expression.*
- template<Expression Ex>  
constexpr auto [erf](#) (Ex const &ex) noexcept  
*Computes Erf of the given expression.*
- template<Expression Ex>  
constexpr auto [erfc](#) (Ex const &ex) noexcept  
*Computes Erfc of the given expression.*
- template<Expression Ex>  
constexpr auto [exp](#) (Ex const &ex) noexcept  
*Computes Exp of the given expression.*
- template<Expression Ex>  
constexpr auto [exp2](#) (Ex const &ex) noexcept  
*Computes Exp2 of the given expression.*

- `template<Expression Ex>`  
`constexpr auto expm1 (Ex const &ex) noexcept`  
*Computes Expm1 of the given expression.*
- `template<Expression Ex>`  
`constexpr auto fabs (Ex const &ex) noexcept`  
*Computes Fabs of the given expression.*
- `template<Expression Ex>`  
`constexpr auto floor (Ex const &ex) noexcept`  
*Computes Floor of the given expression.*
- `template<Expression Ex>`  
`constexpr auto llrint (Ex const &ex) noexcept`  
*Computes Llrint of the given expression.*
- `template<Expression Ex>`  
`constexpr auto llround (Ex const &ex) noexcept`  
*Computes Llround of the given expression.*
- `template<Expression Ex>`  
`constexpr auto log (Ex const &ex) noexcept`  
*Computes Log of the given expression.*
- `template<Expression Ex>`  
`constexpr auto log10 (Ex const &ex) noexcept`  
*Computes Log10 of the given expression.*
- `template<Expression Ex>`  
`constexpr auto log1p (Ex const &ex) noexcept`  
*Computes Log1p of the given expression.*
- `template<Expression Ex>`  
`constexpr auto log2 (Ex const &ex) noexcept`  
*Computes Log2 of the given expression.*
- `template<Expression Ex>`  
`constexpr auto lrint (Ex const &ex) noexcept`  
*Computes Lrint of the given expression.*
- `template<Expression Ex>`  
`constexpr auto lround (Ex const &ex) noexcept`  
*Computes Lround of the given expression.*
- `template<Expression Ex>`  
`constexpr auto nearbyint (Ex const &ex) noexcept`  
*Computes Nearbyint of the given expression.*
- `template<Expression Ex>`  
`constexpr auto rint (Ex const &ex) noexcept`  
*Computes Rint of the given expression.*
- `template<Expression Ex>`  
`constexpr auto round (Ex const &ex) noexcept`  
*Computes Round of the given expression.*
- `template<Expression Ex>`  
`constexpr auto sin (Ex const &ex) noexcept`  
*Computes Sin of the given expression.*
- `template<Expression Ex>`  
`constexpr auto sinh (Ex const &ex) noexcept`  
*Computes Sinh of the given expression.*
- `template<Expression Ex>`  
`constexpr auto sqrt (Ex const &ex) noexcept`  
*Computes Sqrt of the given expression.*
- `template<Expression Ex>`  
`constexpr auto tan (Ex const &ex) noexcept`

*Computes Tan of the given expression.*

- `template<Expression Ex>`  
`constexpr auto tanh (Ex const &ex) noexcept`

*Computes Tanh of the given expression.*

- `template<Expression Ex>`  
`constexpr auto trunc (Ex const &ex) noexcept`

*Computes Trunc of the given expression.*

## Variables

- `static constexpr auto ceras::make\_unary\_operator`
- `static constexpr auto ceras::make\_binary\_operator`
- `template<class T >`  
`constexpr bool ceras::is\_unary\_operator\_v = is_unary_operator<T>::value`
- `template<typename T >`  
`concept ceras::Unary\_Operator = is_unary_operator_v<T>`  
*A type that represents an unary operator.*
- `template<class T >`  
`constexpr bool ceras::is\_binary\_operator\_v = is_binary_operator<T>::value`
- `template<typename T >`  
`concept ceras::Binary\_Operator = is_binary_operator_v<T>`  
*A type that represents a binary operator.*
- `template<typename T >`  
`concept ceras::Operator = Unary_Operator<T> || Binary_Operator<T>`  
*A type that represents an unary or a binary operator.*
- `template<typename T >`  
`concept ceras::Expression = Operator<T> || Variable<T> || Place_Holder<T> || Constant<T> || Value<T>`  
*A type that represents a unary operator, a binary operator, a variable, a [place\\_holder](#), a constant or a value.*
- `*auto y = variable<tensor<float>>{ }`
- `*auto sqr = hypot( x, y )`

## 7.11.1 Function Documentation

### 7.11.1.1 [abs\(\)](#)

```
template<Expression Ex>
constexpr auto abs (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Abs of the given expression.

Example code:

```
auto a = variable{ random<float>({ 2, 3, 5 } ) };
auto b = abs( a );
```

### 7.11.1.2 acos()

```
template<Expression Ex>
constexpr auto acos (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Acos of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = acos( a );
```

### 7.11.1.3 acosh()

```
template<Expression Ex>
constexpr auto acosh (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Acosh of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = acosh( a );
```

### 7.11.1.4 asin()

```
template<Expression Ex>
constexpr auto asin (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Asin of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = asin( a );
```

### 7.11.1.5 asinh()

```
template<Expression Ex>
constexpr auto asinh (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Asinh of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = asinh( a );
```

### 7.11.1.6 atan()

```
template<Expression Ex>
constexpr auto atan (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Atan of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = atan( a );
```

### 7.11.1.7 atan2()

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto atan2 (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

Computes the arc tangent of y/x using the signs of arguments to determine the correct quadrant.

### 7.11.1.8 atanh()

```
template<Expression Ex>
constexpr auto atanh (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Atanh of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = atanh( a );
```

### 7.11.1.9 average\_pooling\_2d()

```
auto average_pooling_2d (
    unsigned long stride ) [inline], [noexcept]
```

### 7.11.1.10 batch\_normalization()

```
template<typename T >
requires std::floating_point<T> auto batch_normalization (
    T const momentum = 0.98 ) [inline], [noexcept]
```

### 7.11.1.11 cbrt()

```
template<Expression Ex>
constexpr auto cbrt (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Cbert of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = cbrt( a );
```

### 7.11.1.12 ceil()

```
template<Expression Ex>
constexpr auto ceil (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Ceil of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = ceil( a );
```

### 7.11.1.13 clip()

```
template<typename Float >
requires std::floating_point<Float> constexpr auto clip (
    Float lower,
    Float upper = std::numeric_limits<Float>::max() ) [constexpr], [noexcept]
```

### 7.11.1.14 concat() [1/2]

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto concat (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

### 7.11.1.15 concat() [2/2]

```
auto concat (
    unsigned long axe = -1 ) [inline]
```

**7.11.1.16 concatenate() [1/2]**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto concatenate (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

**7.11.1.17 concatenate() [2/2]**

```
auto concatenate (
    unsigned long axe = -1 ) [inline]
```

**7.11.1.18 conv2d()**

```
auto conv2d (
    unsigned long row_input,
    unsigned long col_input,
    unsigned long const row_stride = 1,
    unsigned long const col_stride = 1,
    unsigned long const row_dilation = 1,
    unsigned long const col_dilation = 1,
    std::string const & padding = "valid" ) [inline], [noexcept]
```

**7.11.1.19 cos()**

```
template<Expression Ex>
constexpr auto cos (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Cos of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = cos( a );
```

**7.11.1.20 cosh()**

```
template<Expression Ex>
constexpr auto cosh (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Cosh of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = cosh( a );
```



### 7.11.1.21 drop\_out()

```
template<typename T >
requires std::floating_point<T> auto drop_out (
    T const factor ) [inline], [noexcept]
```

### 7.11.1.22 equal()

```
template<Expression Lhs_Expression, Expression Rhs_Expression, std::floating_point FP>
constexpr auto equal (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex,
    FP threshold = 0.5 ) [constexpr], [noexcept]
```

Returns the truth value of (lhs == rhs) element-wise. [+1 for true, 0 for false]

#### Parameters

<i>lhs_ex</i>	The first operator.
<i>rhs_ex</i>	The second operator.

#### Returns

An instance of a binary operator that evaluate the element-wise equality of two input operators.

#### Example code:

```
auto l = variable<tensor<float>>{ /*...*/ };
auto r = place_holder<tensor<float>>{};
auto eq = equal(l, r);
```

### 7.11.1.23 erf()

```
template<Expression Ex>
constexpr auto erf (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Erf of the given expression.

#### Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = erf( a );
```

#### 7.11.1.24 `erfc()`

```
template<Expression Ex>
constexpr auto erfc (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Erfc of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = erfc( a );
```

#### 7.11.1.25 `exp()`

```
template<Expression Ex>
constexpr auto exp (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Exp of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = exp( a );
```

#### 7.11.1.26 `exp2()`

```
template<Expression Ex>
constexpr auto exp2 (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Exp2 of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = exp2( a );
```

#### 7.11.1.27 `expm1()`

```
template<Expression Ex>
constexpr auto expm1 (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Expm1 of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = expm1( a );
```

### 7.11.1.28 fabs()

```
template<Expression Ex>
constexpr auto fabs (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Fabs of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = fabs( a );
```

### 7.11.1.29 flatten()

```
template<Expression Ex>
constexpr auto flatten (
    Ex const & ex ) [constexpr], [noexcept]
```

### 7.11.1.30 floor()

```
template<Expression Ex>
constexpr auto floor (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Floor of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = floor( a );
```

### 7.11.1.31 hypot()

```
template<Expression Ex, Expression Ey>
* endcode* * constexpr auto hypot (
    Ex const & ex,
    Ey const & ey ) [constexpr], [noexcept]
```

### 7.11.1.32 identity()

```
template<Expression Ex>
constexpr auto identity (
    Ex const & ex ) [constexpr], [noexcept]
```

### 7.11.1.33 img2col()

```
auto img2col (
    unsigned long const row_kernel,
    unsigned long col_kernel = -1,
    unsigned long const row_padding = 0,
    unsigned long col_padding = 0,
    unsigned long const row_stride = 1,
    unsigned long const col_stride = 1,
    unsigned long const row_dilation = 1,
    unsigned long const col_dilation = 1 ) [inline], [noexcept]
```

### 7.11.1.34 llrint()

```
template<Expression Ex>
constexpr auto llrint (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Llrint of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = llrint( a );
```

### 7.11.1.35 llround()

```
template<Expression Ex>
constexpr auto llround (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Llround of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = llround( a );
```

### 7.11.1.36 log()

```
template<Expression Ex>
constexpr auto log (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Log of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = log( a );
```

### 7.11.1.37 log10()

```
template<Expression Ex>
constexpr auto log10 (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Log10 of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = log10( a );
```

### 7.11.1.38 log1p()

```
template<Expression Ex>
constexpr auto log1p (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Log1p of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = log1p( a );
```

### 7.11.1.39 log2()

```
template<Expression Ex>
constexpr auto log2 (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Log2 of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = log2( a );
```

### 7.11.1.40 lrint()

```
template<Expression Ex>
constexpr auto lrint (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Lrint of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = lrint( a );
```

#### 7.11.1.41 lround()

```
template<Expression Ex>
constexpr auto lround (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Lround of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = lround( a );
```

#### 7.11.1.42 max\_pooling\_2d()

```
auto max_pooling_2d (
    unsigned long stride ) [inline], [noexcept]
```

#### 7.11.1.43 maximum()

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto maximum (
    Lhs_Expression const & lhs_ex,
    Rhs_Expression const & rhs_ex ) [constexpr], [noexcept]
```

#### 7.11.1.44 nearbyint()

```
template<Expression Ex>
constexpr auto nearbyint (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Nearbyint of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = nearbyint( a );
```

#### 7.11.1.45 normalization\_batch()

```
template<typename T = double>
requires std::floating_point<T> auto normalization_batch (
    T const momentum = 0.98 ) [inline], [noexcept]
```

#### 7.11.1.46 ones\_like()

```
template<Expression Ex>
auto ones_like (
    Ex const & ex ) [noexcept]
```

`ones_like` produces a tensor of the same shape as the input expression, but with every element to be 1.

##### Returns

An unary operator that takes an unary operator, and producing an output tensor Example Code:

```
auto va = variable{ ones<float>({3, 3, 3}) };
auto v_rand = ones_like( va ); // this expression will produces a tensor of shape (3, 3, 3), with every
    element to be 1.
```

#### 7.11.1.47 random\_normal\_like()

```
template<typename T = float>
requires std::floating_point<T> auto random_normal_like (
    T mean = 0.0,
    T stddev = 1.0 ) [inline], [noexcept]
```

`random_normal_like` produces random tensor from a normal distribution

##### Parameters

<i>mean</i>	Mean of the normal distribution, a scalar.
<i>stddev</i>	Standard deviation of the normal distribution, a scalar.

##### Returns

An unary operator that takes an unary operator, and producing output tensor from a normal distribution. The shape of the output tensor has the same shape corresponding to the input unary operator.

##### Example Code

```
auto va = variable{ ones<float>({3, 3, 3}) };
auto v_rand = random_normal_like( 1.0, 4.0 )( va ); // this expression will produces a tensor of shape (3,
    3, 3) from a normal distribution with parameters (1.0, 4.0)
```

#### 7.11.1.48 reduce\_max()

```
auto reduce_max (
    unsigned long axis = -1 ) [inline], [noexcept]
```

Reduce maximum elements along an axis.

##### Parameters

<i>axis</i>	The axis along which to reduce maximum values. Defaults to the last axis.
-------------	---

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = reduce_max( 0 )( a ); // <- output shape is ( 3, 5 )
auto b = reduce_max( 1 )( a ); // <- output shape is ( 2, 5 )
auto b = reduce_max( 2 )( a ); // <- output shape is ( 2, 3 )
auto b = reduce_max( ) ( a ); // <- output shape is ( 2, 3 )
```

#### 7.11.1.49 reduce\_min()

```
auto reduce_min (
    unsigned long axis = -1 ) [inline], [noexcept]
```

Reduce minimal elements along an axis.

Parameters

<i>axis</i>	The axis along which to reduce minimal values. Defaults to the last axis.
-------------	---

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = reduce_min( 0 )( a ); // <- output shape is ( 3, 5 )
auto b = reduce_min( 1 )( a ); // <- output shape is ( 2, 5 )
auto b = reduce_min( 2 )( a ); // <- output shape is ( 2, 3 )
auto b = reduce_min( ) ( a ); // <- output shape is ( 2, 3 )
```

#### 7.11.1.50 reduce\_sum()

```
auto reduce_sum (
    unsigned long axis ) [inline], [noexcept]
```

Reduce sum elements along an axis.

Parameters

<i>axis</i>	The axis along which to reduce sum.
-------------	-------------------------------------

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = reduce_sum( 0 )( a ); // <- output shape is ( 3, 5 )
auto b = reduce_sum( 1 )( a ); // <- output shape is ( 2, 5 )
auto b = reduce_sum( 2 )( a ); // <- output shape is ( 2, 3 )
auto b = reduce_sum( -1 )( a ); // <- output shape is ( 2, 3 )
```

#### 7.11.1.51 repeat()

```
auto repeat (
    unsigned long repeats,
    unsigned long axis = -1 ) [inline], [noexcept]
```

Repeats elements along an axis.



## Parameters

<i>repeats</i>	The number of repetitions for each element.
<i>axis</i>	The axis along which to repeat values. Defaults to the last axis.

## Example code:

```

auto a = variable{ random<float>( {2, 3, 5} ) };
auto b0 = repeat( 2, 0 )( a ); // <- output shape is ( 4, 3, 5 )
auto b1 = repeat( 2, 1 )( a ); // <- output shape is ( 2, 6, 5 )
auto b2 = repeat( 2, 2 )( a ); // <- output shape is ( 2, 3, 10 )
auto bx = repeat( 2 )( a ); // <- output shape is ( 2, 3, 10 )

```

**7.11.1.52 reshape()**

```

auto reshape (
    std::vector< unsigned long > const & new_shape,
    bool include_batch_flag = true ) [inline], [noexcept]

```

**7.11.1.53 rint()**

```

template<Expression Ex>
constexpr auto rint (
    Ex const & ex ) [constexpr], [noexcept]

```

Computes Rint of the given expression.

## Example code:

```

auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = rint( a );

```

**7.11.1.54 round()**

```

template<Expression Ex>
constexpr auto round (
    Ex const & ex ) [constexpr], [noexcept]

```

Computes Round of the given expression.

## Example code:

```

auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = round( a );

```

**7.11.1.55 sign()**

```

template<Expression Ex>
constexpr auto sign (
    Ex const & ex ) [constexpr], [noexcept]

```

Returns the sign. [1 for positive, 0 for 0 and -1 for negative]

**Parameters**

<i>ex</i>	The input operator.
-----------	---------------------

**Returns**

An instance of a unary\_operator that evaluate the sign of the input operator.

**Example code:**

```
auto e = variable<tensor<float>>( /*...*/ );
auto si = sign(e);
```

**7.11.1.56 sin()**

```
template<Expression Ex>
constexpr auto sin (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Sin of the given expression.

**Example code:**

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = sin( a );
```

**7.11.1.57 sinh()**

```
template<Expression Ex>
constexpr auto sinh (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Sinh of the given expression.

**Example code:**

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = sinh( a );
```

**7.11.1.58 sqrt()**

```
template<Expression Ex>
constexpr auto sqrt (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Sqrt of the given expression.

**Example code:**

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = sqrt( a );
```

#### 7.11.1.59 tan()

```
template<Expression Ex>
constexpr auto tan (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Tan of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = tan( a );
```

#### 7.11.1.60 tanh()

```
template<Expression Ex>
constexpr auto tanh (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Tanh of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = tanh( a );
```

#### 7.11.1.61 transpose()

```
template<Expression Ex>
auto transpose (
    Ex const & ex ) [noexcept]
```

#### 7.11.1.62 trunc()

```
template<Expression Ex>
constexpr auto trunc (
    Ex const & ex ) [constexpr], [noexcept]
```

Computes Trunc of the given expression.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = trunc( a );
```

#### 7.11.1.63 up\_sampling\_2d()

```
auto up_sampling_2d (
    unsigned long stride ) [inline], [noexcept]
```

#### 7.11.1.64 zero\_padding\_2d()

```
auto zero_padding_2d (
    std::vector< unsigned long > const & padding ) [inline], [noexcept]
```

Zero-padding layer for 2D input. The input should have 4-dimensions: (batch\_size, row, col, channel). The output has 4-dimensions: (batch\_size, new\_row, new\_col, channel).

## Parameters

<i>padding</i>	If a single integer, then apply symmetric padding to height and width. If two integers, then first is for height and the second is for width. If four integers, then is interpreted as<tt>(top_pad, bottom_pad, left_pad, right_pad).
----------------	---

## Example code:

```
auto a = variable{ random<float>({16, 16, 3}) };
auto b = zero_padding_2d( {8,} )( a ); // shape for b is (8+16+8, 8+16+8, 3)
auto c = zero_padding_2d( {8, 4} )( a ); // shape for c is (8+16+8, 4+16+4, 3)
auto d = zero_padding_2d( {8, 4, 2, 1} )( a ); // shape for d is (8+16+4, 2+16+1, 3)
```

## 7.11.1.65 zeros\_like()

```
template<Expression Ex>
auto zeros_like (
    Ex const & ex ) [noexcept]
```

`zeros_like` produces a tensor of the same shape as the input expression, but with every element to be 0.

## Returns

An unary operator that takes an unary operator, and producing an output tensor Example Code:

```
auto va = variable{ ones<float>({3, 3, 3}) };
auto v_rand = zeros_like( va ); // this expression will produces a tensor of shape (3, 3, 3), with
every element to be 0.
```

## 7.11.2 Variable Documentation

## 7.11.2.1 sqr

```
* auto sqr = hypot( x, y )
```

## 7.11.2.2 y

```
* auto y = variable<tensor<float>>{ }
```

## 7.12 /home/feng/workspace/github.repo/ceras/include/optimizer.hpp File Reference

```
#include " ./config.hpp"
#include " ./operation.hpp"
#include " ./place_holder.hpp"
#include " ./variable.hpp"
#include " ./session.hpp"
#include " ./utils/color.hpp"
#include " ./utils/debug.hpp"
#include " ./utils/id.hpp"
#include " ./utils/enable_shared.hpp"
```

## Classes

- struct [ceras::sgd< Loss, T >](#)
- struct [ceras::adagrad< Loss, T >](#)
- struct [ceras::rmsprop< Loss, T >](#)
- struct [ceras::adadelta< Loss, T >](#)
- struct [ceras::adam< Loss, T >](#)
- struct [ceras::gradient\\_descent< Loss, T >](#)

## Namespaces

- [ceras](#)

## Typedefs

- template<typename Loss , typename T >  
using [ceras::ada\\_grad](#) = adagrad< Loss, T >
- template<typename Loss , typename T >  
using [ceras::rms\\_prop](#) = rmsprop< Loss, T >
- template<typename Loss , typename T >  
using [ceras::ada\\_delta](#) = adadelta< Loss, T >

## Variables

- auto [ceras::Adam](#)
- auto [ceras::SGD](#)
- auto [ceras::Adagrad](#)
- auto [ceras::RMSprop](#)
- auto [ceras::Adadelta](#)

## 7.13 /home/feng/workspace/github.repo/ceras/include/place\_holder.hpp File Reference

```
#include "../includes.hpp"
#include "../tensor.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/debug.hpp"
#include "../utils/id.hpp"
#include "../utils/enable_shared.hpp"
#include "../utils/state.hpp"
```

## Classes

- struct [ceras::place\\_holder\\_state< Tsor >](#)
- struct [ceras::place\\_holder< Tsor >](#)
- struct [ceras::is\\_place\\_holder< T >](#)
- struct [ceras::is\\_place\\_holder< place\\_holder< Tsor > >](#)

## Namespaces

- [ceras](#)

## Functions

- `template<Place_Holder Ph>`  
`bool ceras::operator== (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool ceras::operator!= (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool ceras::operator< (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool ceras::operator> (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool ceras::operator<= (Ph const &lhs, Ph const &rhs)`
- `template<Place_Holder Ph>`  
`bool ceras::operator>= (Ph const &lhs, Ph const &rhs)`

## Variables

- `template<class T >`  
`constexpr bool ceras::is\_place\_holder\_v = is_place_holder<T>::value`
- `template<typename T >`  
`concept ceras::Place\_Holder = is_place_holder_v<T>`

## 7.14 /home/feng/workspace/github.repo/ceras/include/session.hpp File Reference

```
#include "../includes.hpp"
#include "../tensor.hpp"
#include "../place_holder.hpp"
#include "../variable.hpp"
#include "../utils/singleton.hpp"
#include "../utils/debug.hpp"
```

## Classes

- `struct ceras::ceras\_private::session< Tsor >`

## Namespaces

- [ceras](#)
- [ceras::ceras\\_private](#)

## Functions

- `template<Tensor Tsr>`  
`ceras\_private::session< Tsr > & ceras::get\_default\_session ()`

## 7.15 /home/feng/workspace/github.repo/ceras/include/tensor.hpp File Reference

```
#include "../includes.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/range.hpp"
#include "../utils/stride_iterator.hpp"
#include "../utils/for_each.hpp"
#include "../utils/buffered_allocator.hpp"
#include "../utils/debug.hpp"
#include "../utils/id.hpp"
#include "../backend/cuda.hpp"
```

### Classes

- struct [ceras::tensor< T, Allocator >](#)
- struct [ceras::is\\_tensor< T >](#)
- struct [ceras::is\\_tensor< tensor< T, A > >](#)
- struct [ceras::view\\_2d< T >](#)
- struct [ceras::view\\_3d< T >](#)
- struct [ceras::view\\_4d< T >](#)

### Namespaces

- [ceras](#)

### Typedefs

- template<typename T >  
using [ceras::default\\_allocator](#) = std::allocator< T >
- template<typename T >  
using [ceras::matrix](#) = view\_2d< T >
- template<typename T >  
using [ceras::cube](#) = view\_3d< T >
- template<typename T >  
using [ceras::tesseract](#) = view\_4d< T >

### Functions

- template<typename T , typename A = default\_allocator<T>>  
constexpr tensor< T, A > [ceras::as\\_tensor](#) (T val) noexcept
- template<Tensor Tsrc, typename CharT , typename Traits >  
std::basic\_ostream< CharT, Traits > & [ceras::operator<<](#) (std::basic\_ostream< CharT, Traits > &os\_, Tsrc const &tsor)
- template<typename T >  
requires std::floating\_point< T > void [ceras::gemm\\_cpu](#) (T const \*A, bool a\_transposed, T const \*B, bool b\_transposed, unsigned long m, unsigned long n, unsigned long k, T \*C)
- void [ceras::update\\_cuda\\_gemm\\_threshold](#) ()

- `template<typename T>`  
requires `std::floating_point< T >` void `ceras::gemm` (T const \*A, bool a\_transposed, T const \*B, bool b\_transposed, unsigned long m, unsigned long n, unsigned long k, T \*C)
- `template<typename T>`  
requires `std::floating_point< T >` void `ceras::gemm` (view\_2d< T > const &x, view\_2d< T > const &y, view\_2d< T > &ans)
- `template<Tensor Tsr>`  
Tsr `ceras::add` (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator+` (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator+` (typename Tsr::value\_type const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator+` (Tsr const &lhs, typename Tsr::value\_type const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::minus` (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator-` (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator-` (typename Tsr::value\_type const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator-` (Tsr const &lhs, typename Tsr::value\_type const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator*` (typename Tsr::value\_type const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator*` (Tsr const &lhs, typename Tsr::value\_type const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator/` (Tsr const &lhs, typename Tsr::value\_type const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::reshape` (Tsr const &ts, std::vector< unsigned long > const &new\_shape)
- `template<Tensor Tsr>`  
void `ceras::multiply` (Tsr const &lhs, Tsr const &rhs, Tsr &ans) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::multiply` (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::operator*` (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::elementwise_product` (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::hadamard_product` (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::elementwise_divide` (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::repeat` (Tsr const &tsor, unsigned long n)
- `template<Tensor Tsr>`  
Tsr `ceras::reduce_sum` (Tsr const &tsor)
- `template<Tensor Tsr>`  
Tsr `ceras::reduce_mean` (Tsr const &tsor)
- `template<Tensor Tsr>`  
Tsr `ceras::clip` (Tsr &tsor, typename Tsr::value\_type lower=0, typename Tsr::value\_type upper=1)
- `template<Tensor Tsr>`  
Tsr `ceras::squeeze` (Tsr const &tsor)
- `template<typename T, typename A = default_allocator<T>>`  
tensor< T, A > `ceras::randn` (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1})
- `template<typename T, typename A = default_allocator<T>>`  
tensor< T, A > `ceras::truncated_normal` (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1}, T lower=T{0}, T upper=T{1})



- `template<typename T, typename A = default_allocator<T>>`  
`tensor< T, A > ceras::random (std::vector< unsigned long > const &shape, T min=T{0}, T max=T{1})`
- `template<Tensor Tsr>`  
`Tsr ceras::random\_like (Tsr const &tsr, typename Tsr::value_type min=0, typename Tsr::value_type max=1)`
- `template<Tensor Tsr>`  
`Tsr ceras::randn\_like (Tsr const &tsr, typename Tsr::value_type mean=0, typename Tsr::value_type stddev=1)`
- `template<typename T, typename A = default_allocator<T>>`  
`tensor< T, A > ceras::glorot\_uniform (std::initializer_list< unsigned long > shape)`
- `template<Tensor Tsr>`  
`Tsr ceras::deep\_copy (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`Tsr ceras::copy (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`Tsr ceras::concatenate (Tsr const &lhs, Tsr const &rhs, unsigned long axis=0) noexcept`
- `template<Tensor Tsr>`  
`Tsr ceras::repmat (Tsr const &tsr, unsigned long row_rep, unsigned long col_rep)`
- `template<Tensor Tsr>`  
`constexpr bool ceras::empty (Tsr const &tsr) noexcept`
- `template<typename T, typename A = default_allocator<T>>`  
`constexpr tensor< T, A > ceras::zeros (std::vector< unsigned long > const &shape)`
- `template<Tensor Tsr>`  
`constexpr Tsr ceras::zeros\_like (Tsr const &tsr)`
- `template<typename T, typename A = default_allocator<T>>`  
`constexpr tensor< T, A > ceras::ones (std::vector< unsigned long > const &shape)`
- `template<Tensor Tsr>`  
`constexpr Tsr ceras::ones\_like (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::max (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::amax (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::min (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::amin (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::sum (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::mean (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::norm (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`Tsr ceras::abs (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`Tsr ceras::softmax (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`bool ceras::has\_nan (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`bool ceras::has\_inf (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`bool ceras::is\_valid (Tsr const &tsr)`
- `template<Tensor Tsr, typename Function >`  
`Tsr ceras::reduce (Tsr const &ts, unsigned long axis, typename Tsr::value_type const &init, Function const &func, bool keepdims=false) noexcept`
- `template<Tensor Tsr>`  
`Tsr ceras::sum (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept`

- `template<Tensor Tsr>`  
requires `std::floating_point< typename Tsr::value_type > Tsr` `ceras::mean` (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept
- `template<Tensor Tsr>`  
requires `std::floating_point< typename Tsr::value_type > Tsr` `ceras::variance` (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept
- `template<Tensor Tsr>`  
requires `std::floating_point< typename Tsr::value_type > Tsr` `ceras::standard_deviation` (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept
- `template<Tensor Tsr>`  
requires `std::floating_point< typename Tsr::value_type > Tsr::value_type` `ceras::var` (Tsr const &ts) noexcept
- `template<Tensor Tsr>`  
requires `std::floating_point< typename Tsr::value_type > Tsr::value_type` `ceras::std` (Tsr const &ts) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::max` (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept
- `template<Tensor Tsr>`  
Tsr `ceras::min` (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept
- `template<typename T, typename A = default_allocator<T>>`  
requires `std::floating_point< T > tensor< T, A >` `ceras::linspace` (T start, T stop, unsigned long num, bool endpoint=true) noexcept
- `template<class _Tp, class _CharT, class _Traits, class _Alloc >`  
`std::basic_istream< _CharT, _Traits > & ceras::read_tensor` (`std::basic_istream< _CharT, _Traits > &__is`, `tensor< _Tp, _Alloc > &__x`)
- `template<class _Tp, class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & ceras::write_tensor` (`std::basic_ostream< _CharT, _Traits > &__os`, `tensor< _Tp, _Alloc > const &__x`)
- `template<typename T, typename A = default_allocator<T>>`  
`tensor< T, A > ceras::load_tensor` (`std::string const &file_name`)
- `template<Tensor Tsr>`  
void `ceras::save_tensor` (`std::string const &file_name`, Tsr const &tsr)

## Variables

- static unsigned long `ceras::random_seed` = `std::chrono::system_clock::now().time_since_epoch().count()`
- static `std::mt19937` `ceras::random_generator` {`random_seed`}
- `template<class T >`  
`constexpr bool ceras::is_tensor_v` = `is_tensor<T>::value`
- `template<typename T >`  
concept `ceras::Tensor` = `is_tensor_v<T>`

## 7.16 /home/feng/workspace/github.repo/ceras/include/value.hpp File Reference

```
#include "../includes.hpp"
#include "../tensor.hpp"
#include "../utils/id.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/enable_shared.hpp"
```

## Classes

- struct [ceras::value< T >](#)
- struct [ceras::is\\_value< T >](#)
- struct [ceras::is\\_value< value< T > >](#)
- struct [ceras::tensor\\_deduction< L, R >](#)

## Namespaces

- [ceras](#)

## Variables

- template<class T >  
constexpr bool [ceras::is\\_value\\_v](#) = is\_value<T>::value
- template<typename T >  
concept [ceras::Value](#) = is\_value\_v<T>

## 7.17 /home/feng/workspace/github.repo/ceras/include/variable.hpp File Reference

```
#include "../includes.hpp"
#include "../tensor.hpp"
#include "../utils/id.hpp"
#include "../utils/debug.hpp"
#include "../config.hpp"
#include "../utils/enable_shared.hpp"
#include "../utils/state.hpp"
```

## Classes

- struct [ceras::variable\\_state< Tsor >](#)
- struct [ceras::regularizer< Float >](#)
- struct [ceras::variable< Tsor >](#)
- struct [ceras::is\\_variable< T >](#)
- struct [ceras::is\\_variable< variable< Tsor > >](#)

## Namespaces

- [ceras](#)
- [ceras::ceras\\_private](#)

## Functions

- template<Tensor Tsor>  
[ceras\\_private::session< Tsor >](#) & [ceras::get\\_default\\_session](#) ()
- template<Variable Var>  
bool [ceras::operator==](#) (Var const &lhs, Var const &rhs) noexcept

## Variables

- template<class T >  
constexpr bool [ceras::is\\_variable\\_v](#) = is\_variable<T>::value
- template<typename T >  
concept [ceras::Variable](#) = is\_variable\_v<T>



# Index

/home/feng/workspace/github.repo/ceras/include/activation.hpp, 139  
Adadelta  
ceras, 53  
/home/feng/workspace/github.repo/ceras/include/ceras.hpp, 140  
adadelta  
ceras::adadelta< Loss, T >, 64  
/home/feng/workspace/github.repo/ceras/include/complex\_operation.hpp, 140  
Adagrad  
ceras, 53  
/home/feng/workspace/github.repo/ceras/include/config.hpp, 142  
adagrad  
ceras::adagrad< Loss, T >, 66  
/home/feng/workspace/github.repo/ceras/include/constant.hpp, 142  
Adam  
ceras, 53  
/home/feng/workspace/github.repo/ceras/include/dataset.hpp, 143  
adam  
ceras::adam< Loss, T >, 68  
/home/feng/workspace/github.repo/ceras/include/includes.hpp, 143  
Add  
ceras, 20  
/home/feng/workspace/github.repo/ceras/include/layer.hpp, 144  
add  
ceras, 20  
/home/feng/workspace/github.repo/ceras/include/loss.hpp, 145  
allocator  
ceras::tensor< T, Allocator >, 107  
/home/feng/workspace/github.repo/ceras/include/model.hpp, 147  
amax  
ceras, 20  
/home/feng/workspace/github.repo/ceras/include/operation.hpp, 147  
asin  
ceras, 21  
/home/feng/workspace/github.repo/ceras/include/optimizer.hpp, 168  
rmsgrad  
ceras::adam< Loss, T >, 68  
/home/feng/workspace/github.repo/ceras/include/place\_holder.hpp, 169  
arg  
ceras, 21  
/home/feng/workspace/github.repo/ceras/include/session.hpp, 170  
as\_scalar  
ceras::tensor< T, Allocator >, 109  
/home/feng/workspace/github.repo/ceras/include/tensor.hpp, 171  
as\_tensor  
ceras, 21  
/home/feng/workspace/github.repo/ceras/include/value.hpp, 174  
as\_type  
ceras::tensor< T, Allocator >, 109  
/home/feng/workspace/github.repo/ceras/include/variable.hpp, 175  
asin  
operation.hpp, 153  
~session  
asinh  
operation.hpp, 153  
ceras::ceras\_private::session< T, Allocator >, 100  
atan  
operation.hpp, 153  
abs  
atan2  
operation.hpp, 154  
ceras, 19, 20  
atanh  
operation.hpp, 154  
operation.hpp, 152  
average\_pooling\_2d  
operation.hpp, 154  
abs\_loss  
ceras, 20  
acos  
AveragePooling2D  
ceras, 21  
acosh  
backward  
ceras::binary\_operator< Lhs\_Operator, Rh\_Operator, Forward\_Action, Backward\_Action >, 71  
ada\_delta  
ceras, 18  
ada\_grad  
ceras, 18

- ceras::constant< Tzor >, 79
  - ceras::place\_holder< Tzor >, 93
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 117
  - ceras::value< T >, 120
  - ceras::variable< Tzor >, 123
- backward\_action\_
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 71
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 118
- batch\_normalization
  - operation.hpp, 154
- batch\_size\_
  - ceras::view\_4d< T >, 138
- BatchNormalization
  - ceras, 21, 22
- begin
  - ceras::tensor< T, Allocator >, 109
  - ceras::view\_2d< T >, 129
- beta\_1\_
  - ceras::adam< Loss, T >, 68
- beta\_2\_
  - ceras::adam< Loss, T >, 69
- binary\_cross\_entropy\_loss
  - ceras, 22
- Binary\_Operator
  - ceras, 53
- binary\_operator
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 70
- BinaryCrossEntropy
  - ceras, 54
- BinaryCrossentropy
  - ceras, 54
- bind
  - ceras::ceras\_private::session< Tzor >, 101
  - ceras::place\_holder< Tzor >, 93
- CategoricalCrossEntropy
  - ceras, 54
- CategoricalCrossentropy
  - ceras, 54
- cbegin
  - ceras::tensor< T, Allocator >, 110
- cbrt
  - operation.hpp, 154
- ceil
  - operation.hpp, 155
- cend
  - ceras::tensor< T, Allocator >, 110
- ceras, 9
  - abs, 19, 20
  - abs\_loss, 20
  - ada\_delta, 18
  - ada\_grad, 18
  - Adadelata, 53
  - Adagrad, 53
  - Adam, 53
  - Add, 20
  - add, 20
  - amax, 20
  - amin, 21
  - arg, 21
  - as\_tensor, 21
  - AveragePooling2D, 21
  - BatchNormalization, 21, 22
  - binary\_cross\_entropy\_loss, 22
  - Binary\_Operator, 53
  - BinaryCrossEntropy, 54
  - BinaryCrossentropy, 54
  - CategoricalCrossEntropy, 54
  - CategoricalCrossentropy, 54
  - clip, 22
  - Complex, 55
  - computation\_graph, 23
  - Concatenate, 23
  - concatenate, 23
  - conj, 24
  - Constant, 55
  - Conv2D, 24
  - copy, 25
  - crelu, 25
  - cross\_entropy, 25
  - cross\_entropy\_loss, 25
  - cube, 18
  - deep\_copy, 26
  - default\_allocator, 19
  - Dense, 26
  - Dropout, 26
  - elementwise\_divide, 27
  - elementwise\_multiply, 27
  - elementwise\_product, 27
  - ELU, 27
  - elu, 27
  - empty, 28
  - exponential, 28
  - Expression, 55
  - Flatten, 28
  - gelu, 28
  - gemm, 28
  - gemm\_cpu, 29
  - get\_default\_session, 29
  - glorot\_uniform, 29
  - hadamard\_product, 29
  - hard\_sigmoid, 30
  - has\_inf, 30
  - has\_nan, 30
  - Hinge, 55
  - hinge\_loss, 30
  - imag, 30
  - Input, 31
  - is\_binary\_operator\_v, 55
  - is\_complex\_v, 56
  - is\_constant\_v, 56
  - is\_place\_holder\_v, 56
  - is\_tensor\_v, 56

is\_unary\_operator\_v, 56  
is\_valid, 31  
is\_value\_v, 56  
is\_variable\_v, 56  
leaky\_relu, 31  
LeakyReLU, 31  
linspace, 31  
load\_tensor, 31  
MAE, 57  
mae, 32  
make\_binary\_operator, 57  
make\_compiled\_model, 32  
make\_trainable, 32  
make\_unary\_operator, 57  
matrix, 19  
max, 32  
MaxPooling2D, 33  
mean, 33  
mean\_absolute\_error, 33  
mean\_reduce, 33  
mean\_squared\_error, 34  
mean\_squared\_logarithmic\_error, 34  
MeanAbsoluteError, 57  
MeanSquaredError, 58  
min, 34  
minus, 34, 35  
MSE, 58  
mse, 35  
Multiply, 35  
multiply, 35  
negative, 36  
negative\_relu, 36  
norm, 36  
ones, 36  
ones\_like, 37  
Operator, 58  
operator!=, 37  
operator<, 42  
operator<<, 42  
operator<=, 42  
operator>, 43  
operator>=, 43  
operator\*, 37, 38  
operator+, 38–40  
operator-, 40–42  
operator/, 42  
operator==, 43  
Place\_Holder, 59  
plus, 43  
polar, 43  
randn, 44  
randn\_like, 44  
random, 44  
random\_generator, 59  
random\_like, 44  
random\_seed, 59  
read\_tensor, 45  
real, 45  
reduce, 45  
reduce\_mean, 45  
reduce\_sum, 46  
ReLU, 46  
relu, 46  
relu6, 46  
repeat, 46  
replace\_placeholder\_with\_expression, 47  
repmat, 47  
Reshape, 47  
reshape, 47  
rms\_prop, 19  
RMSprop, 59  
save\_tensor, 48  
selu, 48  
SGD, 59  
sigmoid, 48  
silu, 48  
Softmax, 48  
softmax, 48, 49  
softplus, 49  
softsign, 49  
square, 49  
squared\_loss, 50  
squeeze, 50  
standard\_deviation, 50  
std, 50  
Subtract, 50  
sum, 50, 51  
sum\_reduce, 51  
swish, 51  
Tensor, 59  
tesseract, 19  
truncated\_normal, 51  
Unary\_Operator, 60  
update\_cuda\_gemm\_threshold, 52  
UpSampling2D, 52  
Value, 60  
var, 52  
Variable, 60  
variance, 52  
write\_tensor, 52  
zeros, 52  
zeros\_like, 53  
ceras::adadelta< Loss, T >, 63  
    adadelta, 64  
    forward, 64  
    iterations\_, 64  
    learning\_rate\_, 64  
    loss\_, 64  
    rho\_, 65  
    tensor\_type, 63  
ceras::adagrad< Loss, T >, 65  
    adagrad, 66  
    decay\_, 66  
    forward, 66  
    iterations\_, 66  
    learning\_rate\_, 66

- loss\_, 67
- tensor\_type, 65
- ceras::adam< Loss, T >, 67
  - adam, 68
  - amsgrad\_, 68
  - beta\_1\_, 68
  - beta\_2\_, 69
  - forward, 68
  - iterations\_, 69
  - learning\_rate\_, 69
  - loss\_, 69
  - tensor\_type, 68
- ceras::binary\_operator< Lhs\_Operator, Rh\_Operator, Forward\_Action, Backward\_Action >, 69
  - backward, 71
  - backward\_action\_, 71
  - binary\_operator, 70
  - forward, 71
  - forward\_action\_, 71
  - lhs\_input\_data\_, 71
  - lhs\_op\_, 71
  - output\_data\_, 72
  - rhs\_input\_data\_, 72
  - rhs\_op\_, 72
  - tensor\_type, 70
- ceras::ceras\_private, 60
- ceras::ceras\_private::session< T\_Sor >, 99
  - ~session, 100
  - bind, 101
  - deserialize, 101
  - operator=, 101
  - place\_holder\_type, 99
  - place\_holders\_, 103
  - rebind, 101
  - remember, 101
  - restore, 102
  - run, 102
  - save, 102
  - serialize, 102
  - session, 100
  - tap, 102
  - variable\_state\_type, 99
  - variable\_type, 100
  - variables\_, 103
- ceras::compiled\_model< Model, Optimizer, Loss >, 72
  - compiled\_model, 73
  - compiled\_optimizer\_, 76
  - evaluate, 74
  - fit, 74
  - ground\_truth\_place\_holder\_, 76
  - input\_place\_holder\_, 76
  - io\_layer\_type, 73
  - loss\_, 76
  - model\_, 76
  - operator(), 75
  - optimizer\_, 77
  - optimizer\_type, 77
  - predict, 75
  - train\_on\_batch, 75
  - trainable, 76
- ceras::complex< Real\_Ex, Imag\_Ex >, 77
  - imag\_, 77
  - real\_, 77
- ceras::constant< T\_Sor >, 78
  - backward, 79
  - constant, 78
  - data\_, 79
  - forward, 79
  - shape, 79
- ceras::dataset, 60
- ceras::dataset::fashion\_mnist, 60
  - load\_data, 61
- ceras::dataset::mnist, 61
  - load\_data, 61
- ceras::gradient\_descent< Loss, T >, 80
  - forward, 81
  - gradient\_descent, 80
  - learning\_rate\_, 81
  - loss\_, 81
  - momentum\_, 81
  - tensor\_type, 80
- ceras::is\_binary\_operator< binary\_operator< Lhs\_Operator, Rh\_Operator, Forward\_Action, Backward\_Action > >, 82
- ceras::is\_binary\_operator< T >, 81
- ceras::is\_complex< complex< Real\_Ex, Imag\_Ex > >, 82
- ceras::is\_complex< T >, 82
- ceras::is\_constant< constant< T\_Sor > >, 83
- ceras::is\_constant< T >, 83
- ceras::is\_place\_holder< place\_holder< T\_Sor > >, 84
- ceras::is\_place\_holder< T >, 83
- ceras::is\_tensor< T >, 84
- ceras::is\_tensor< tensor< T, A > >, 84
- ceras::is\_unary\_operator< T >, 85
- ceras::is\_unary\_operator< unary\_operator< Operator, Forward\_Action, Backward\_Action > >, 85
- ceras::is\_value< T >, 85
- ceras::is\_value< value< T > >, 86
- ceras::is\_variable< T >, 86
- ceras::is\_variable< variable< T\_Sor > >, 86
- ceras::model< Ex, Ph >, 87
  - compile, 88
  - expression\_, 91
  - input, 89
  - input\_layer\_type, 87
  - load\_weights, 89
  - model, 88
  - operator(), 89
  - output, 90
  - output\_layer\_type, 88
  - place\_holder\_, 91
  - predict, 90
  - save\_weights, 90
  - summary, 90
  - trainable, 91



- ceras::place\_holder< T, S, A >, 92
  - backward, 93
  - bind, 93
  - forward, 93
  - operator=, 93, 94
  - place\_holder, 92, 93
  - reset, 94
  - tensor\_type, 92
- ceras::place\_holder\_state< T, S, A >, 94
  - data\_, 94
  - shape\_hint\_, 94
- ceras::regularizer< T, S, A >, 95
  - l1\_, 96
  - l2\_, 96
  - regularizer, 95
  - synchronized\_, 96
  - value\_type, 95
- ceras::rmsprop< T, S, A >, 96
  - decay\_, 98
  - forward, 97
  - iterations\_, 98
  - learning\_rate\_, 98
  - loss\_, 98
  - rho\_, 98
  - rmsprop, 97
  - tensor\_type, 97
- ceras::sgd< T, S, A >, 103
  - decay\_, 104
  - forward, 104
  - iterations\_, 105
  - learning\_rate\_, 105
  - loss\_, 105
  - momentum\_, 105
  - nesterov\_, 105
  - sgd, 104
  - tensor\_type, 104
- ceras::tensor< T, S, A >, 106
  - allocator, 107
  - as\_scalar, 109
  - as\_type, 109
  - begin, 109
  - cbegin, 110
  - cend, 110
  - copy, 110
  - creep\_to, 110
  - data, 110
  - deep\_copy, 110, 111
  - empty, 111
  - end, 111
  - map, 111
  - memory\_offset\_, 115
  - ndim, 111
  - operator\*=, 112
  - operator+=, 112
  - operator-, 112
  - operator=, 112, 113
  - operator/=, 113
  - operator=, 113
  - operator[], 113, 114
  - reset, 114
  - reshape, 114
  - resize, 114
  - self\_type, 107
  - shape, 114
  - shape\_, 115
  - shared\_vector, 107
  - shrink\_to, 115
  - size, 115
  - slice, 115
  - tensor, 108, 109
  - value\_type, 107
  - vector\_, 115
  - vector\_type, 108
- ceras::tensor\_deduction< L, R >, 116
  - op\_type, 116
  - tensor\_type, 116
- ceras::unary\_operator< T, S, A, Operator, Forward\_Action, Backward\_Action >, 116
  - backward, 117
  - backward\_action\_, 118
  - forward, 117
  - forward\_action\_, 118
  - input\_data\_, 118
  - op\_, 118
  - output\_data\_, 118
  - tensor\_type, 118
  - unary\_operator, 117
- ceras::value< T, S, A >, 119
  - backward, 120
  - data\_, 121
  - forward, 120
  - operator=, 120, 121
  - value, 119, 120
  - value\_type, 119
- ceras::variable< T, S, A >, 121
  - backward, 123
  - contexts, 123
  - data, 124
  - forward, 124
  - gradient, 124
  - operator=, 124
  - regularizer\_, 125
  - reset, 125
  - shape, 125
  - state\_, 125
  - tensor\_type, 122
  - trainable, 125
  - trainable\_, 126
  - value\_type, 122
  - variable, 122, 123
- ceras::variable\_state< T, S, A >, 126
  - contexts\_, 126
  - data\_, 126
  - gradient\_, 126
- ceras::view\_2d< T, S, A >, 127
  - begin, 129

- col, [129](#)
- col\_, [132](#)
- col\_begin, [129](#), [130](#)
- col\_end, [130](#)
- col\_type, [128](#)
- const\_col\_type, [128](#)
- const\_row\_type, [128](#)
- data, [130](#)
- data\_, [132](#)
- end, [130](#)
- operator[], [131](#)
- row, [131](#)
- row\_, [132](#)
- row\_begin, [131](#)
- row\_end, [131](#), [132](#)
- row\_type, [128](#)
- shape, [132](#)
- size, [132](#)
- transposed\_, [133](#)
- value\_type, [128](#)
- view\_2d, [128](#), [129](#)
- ceras::view\_3d< T >, [133](#)
  - channel\_, [134](#)
  - col\_, [134](#)
  - data\_, [134](#)
  - operator[], [134](#)
  - row\_, [134](#)
  - view\_3d, [133](#)
- ceras::view\_4d< T >, [135](#)
  - batch\_size\_, [138](#)
  - channel\_, [138](#)
  - col\_, [138](#)
  - data\_, [138](#)
  - operator[], [137](#)
  - row\_, [138](#)
  - view\_4d, [135](#)
- channel\_
  - ceras::view\_3d< T >, [134](#)
  - ceras::view\_4d< T >, [138](#)
- clip
  - ceras, [22](#)
  - operation.hpp, [155](#)
- col
  - ceras::view\_2d< T >, [129](#)
- col\_
  - ceras::view\_2d< T >, [132](#)
  - ceras::view\_3d< T >, [134](#)
  - ceras::view\_4d< T >, [138](#)
- col\_begin
  - ceras::view\_2d< T >, [129](#), [130](#)
- col\_end
  - ceras::view\_2d< T >, [130](#)
- col\_type
  - ceras::view\_2d< T >, [128](#)
- compile
  - ceras::model< Ex, Ph >, [88](#)
- compiled\_model
  - ceras::compiled\_model< Model, Optimizer, Loss >, [73](#)
- compiled\_optimizer\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, [76](#)
- Complex
  - ceras, [55](#)
- computation\_graph
  - ceras, [23](#)
- concat
  - operation.hpp, [155](#)
- Concatenate
  - ceras, [23](#)
- concatenate
  - ceras, [23](#)
  - operation.hpp, [155](#), [156](#)
- conj
  - ceras, [24](#)
- const\_col\_type
  - ceras::view\_2d< T >, [128](#)
- const\_row\_type
  - ceras::view\_2d< T >, [128](#)
- Constant
  - ceras, [55](#)
- constant
  - ceras::constant< Tzor >, [78](#)
- contexts
  - ceras::variable< Tzor >, [123](#)
- contexts\_
  - ceras::variable\_state< Tzor >, [126](#)
- Conv2D
  - ceras, [24](#)
- conv2d
  - operation.hpp, [156](#)
- copy
  - ceras, [25](#)
  - ceras::tensor< T, Allocator >, [110](#)
- cos
  - operation.hpp, [156](#)
- cosh
  - operation.hpp, [156](#)
- creep\_to
  - ceras::tensor< T, Allocator >, [110](#)
- crelu
  - ceras, [25](#)
- cross\_entropy
  - ceras, [25](#)
- cross\_entropy\_loss
  - ceras, [25](#)
- cube
  - ceras, [18](#)
- data
  - ceras::tensor< T, Allocator >, [110](#)
  - ceras::variable< Tzor >, [124](#)
  - ceras::view\_2d< T >, [130](#)
- data\_
  - ceras::constant< Tzor >, [79](#)
  - ceras::place\_holder\_state< Tzor >, [94](#)

- ceras::value< T >, [121](#)
  - ceras::variable\_state< T, Allocator >, [126](#)
  - ceras::view\_2d< T >, [132](#)
  - ceras::view\_3d< T >, [134](#)
  - ceras::view\_4d< T >, [138](#)
- decay\_
  - ceras::adagrad< Loss, T >, [66](#)
  - ceras::rmsprop< Loss, T >, [98](#)
  - ceras::sgd< Loss, T >, [104](#)
- deep\_copy
  - ceras, [26](#)
  - ceras::tensor< T, Allocator >, [110](#), [111](#)
- default\_allocator
  - ceras, [19](#)
- Dense
  - ceras, [26](#)
- deserialize
  - ceras::ceras\_private::session< T, Allocator >, [101](#)
- drop\_out
  - operation.hpp, [156](#)
- Dropout
  - ceras, [26](#)
- elementwise\_divide
  - ceras, [27](#)
- elementwise\_multiply
  - ceras, [27](#)
- elementwise\_product
  - ceras, [27](#)
- ELU
  - ceras, [27](#)
- elu
  - ceras, [27](#)
- empty
  - ceras, [28](#)
  - ceras::tensor< T, Allocator >, [111](#)
- end
  - ceras::tensor< T, Allocator >, [111](#)
  - ceras::view\_2d< T >, [130](#)
- equal
  - operation.hpp, [157](#)
- erf
  - operation.hpp, [157](#)
- erfc
  - operation.hpp, [157](#)
- evaluate
  - ceras::compiled\_model< Model, Optimizer, Loss >, [74](#)
- exp
  - operation.hpp, [158](#)
- exp2
  - operation.hpp, [158](#)
- expm1
  - operation.hpp, [158](#)
- exponential
  - ceras, [28](#)
- Expression
  - ceras, [55](#)
- expression\_
  - ceras::model< Ex, Ph >, [91](#)
- fabs
  - operation.hpp, [158](#)
- fit
  - ceras::compiled\_model< Model, Optimizer, Loss >, [74](#)
- Flatten
  - ceras, [28](#)
- flatten
  - operation.hpp, [159](#)
- floor
  - operation.hpp, [159](#)
- forward
  - ceras::adadelta< Loss, T >, [64](#)
  - ceras::adagrad< Loss, T >, [66](#)
  - ceras::adam< Loss, T >, [68](#)
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, [71](#)
  - ceras::constant< T, Allocator >, [79](#)
  - ceras::gradient\_descent< Loss, T >, [81](#)
  - ceras::place\_holder< T, Allocator >, [93](#)
  - ceras::rmsprop< Loss, T >, [97](#)
  - ceras::sgd< Loss, T >, [104](#)
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, [117](#)
  - ceras::value< T >, [120](#)
  - ceras::variable< T, Allocator >, [124](#)
- forward\_action\_
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, [71](#)
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, [118](#)
- gelu
  - ceras, [28](#)
- gemm
  - ceras, [28](#)
- gemm\_cpu
  - ceras, [29](#)
- get\_default\_session
  - ceras, [29](#)
- glorot\_uniform
  - ceras, [29](#)
- gradient
  - ceras::variable< T, Allocator >, [124](#)
- gradient\_
  - ceras::variable\_state< T, Allocator >, [126](#)
- gradient\_descent
  - ceras::gradient\_descent< Loss, T >, [80](#)
- ground\_truth\_place\_holder\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, [76](#)
- hadamard\_product
  - ceras, [29](#)
- hard\_sigmoid
  - ceras, [30](#)
- has\_inf

- ceras, 30
- has\_nan
  - ceras, 30
- Hinge
  - ceras, 55
- hinge\_loss
  - ceras, 30
- hypot
  - operation.hpp, 159
- identity
  - operation.hpp, 159
- imag
  - ceras, 30
- imag\_
  - ceras::complex< Real\_Ex, Imag\_Ex >, 77
- img2col
  - operation.hpp, 159
- includes.hpp
  - STB\_IMAGE\_IMPLEMENTATION, 144
  - STB\_IMAGE\_RESIZE\_IMPLEMENTATION, 144
  - STB\_IMAGE\_WRITE\_IMPLEMENTATION, 144
- Input
  - ceras, 31
- input
  - ceras::model< Ex, Ph >, 89
- input\_data\_
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 118
- input\_layer\_type
  - ceras::model< Ex, Ph >, 87
- input\_place\_holder\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, 76
- io\_layer\_type
  - ceras::compiled\_model< Model, Optimizer, Loss >, 73
- is\_binary\_operator\_v
  - ceras, 55
- is\_complex\_v
  - ceras, 56
- is\_constant\_v
  - ceras, 56
- is\_place\_holder\_v
  - ceras, 56
- is\_tensor\_v
  - ceras, 56
- is\_unary\_operator\_v
  - ceras, 56
- is\_valid
  - ceras, 31
- is\_value\_v
  - ceras, 56
- is\_variable\_v
  - ceras, 56
- iterations\_
  - ceras::adadelta< Loss, T >, 64
  - ceras::adagrad< Loss, T >, 66
  - ceras::adam< Loss, T >, 69
- ceras::rmsprop< Loss, T >, 98
- ceras::sgd< Loss, T >, 105
- l1\_
  - ceras::regularizer< Float >, 96
- l2\_
  - ceras::regularizer< Float >, 96
- leaky\_relu
  - ceras, 31
- LeakyReLU
  - ceras, 31
- learning\_rate\_
  - ceras::adadelta< Loss, T >, 64
  - ceras::adagrad< Loss, T >, 66
  - ceras::adam< Loss, T >, 69
  - ceras::gradient\_descent< Loss, T >, 81
  - ceras::rmsprop< Loss, T >, 98
  - ceras::sgd< Loss, T >, 105
- lhs\_input\_data\_
  - ceras::binary\_operator< Lhs\_Operator, Rhc\_Operator, Forward\_Action, Backward\_Action >, 71
- lhs\_op\_
  - ceras::binary\_operator< Lhs\_Operator, Rhc\_Operator, Forward\_Action, Backward\_Action >, 71
- linspace
  - ceras, 31
- llrint
  - operation.hpp, 160
- llround
  - operation.hpp, 160
- load\_data
  - ceras::dataset::fashion\_mnist, 61
  - ceras::dataset::mnist, 61
- load\_tensor
  - ceras, 31
- load\_weights
  - ceras::model< Ex, Ph >, 89
- log
  - operation.hpp, 160
- log10
  - operation.hpp, 160
- log1p
  - operation.hpp, 161
- log2
  - operation.hpp, 161
- loss\_
  - ceras::adadelta< Loss, T >, 64
  - ceras::adagrad< Loss, T >, 67
  - ceras::adam< Loss, T >, 69
  - ceras::compiled\_model< Model, Optimizer, Loss >, 76
  - ceras::gradient\_descent< Loss, T >, 81
  - ceras::rmsprop< Loss, T >, 98
  - ceras::sgd< Loss, T >, 105
- lrint
  - operation.hpp, 161
- lround
  - operation.hpp, 161

- MAE
  - ceras, [57](#)
- mae
  - ceras, [32](#)
- make\_binary\_operator
  - ceras, [57](#)
- make\_compiled\_model
  - ceras, [32](#)
- make\_trainable
  - ceras, [32](#)
- make\_unary\_operator
  - ceras, [57](#)
- map
  - ceras::tensor< T, Allocator >, [111](#)
- matrix
  - ceras, [19](#)
- max
  - ceras, [32](#)
- max\_pooling\_2d
  - operation.hpp, [162](#)
- maximum
  - operation.hpp, [162](#)
- MaxPooling2D
  - ceras, [33](#)
- mean
  - ceras, [33](#)
- mean\_absolute\_error
  - ceras, [33](#)
- mean\_reduce
  - ceras, [33](#)
- mean\_squared\_error
  - ceras, [34](#)
- mean\_squared\_logarithmic\_error
  - ceras, [34](#)
- MeanAbsoluteError
  - ceras, [57](#)
- MeanSquaredError
  - ceras, [58](#)
- memory\_offset\_
  - ceras::tensor< T, Allocator >, [115](#)
- min
  - ceras, [34](#)
- minus
  - ceras, [34](#), [35](#)
- model
  - ceras::model< Ex, Ph >, [88](#)
- model\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, [76](#)
- momentum\_
  - ceras::gradient\_descent< Loss, T >, [81](#)
  - ceras::sgd< Loss, T >, [105](#)
- MSE
  - ceras, [58](#)
- mse
  - ceras, [35](#)
- Multiply
  - ceras, [35](#)
- multiply
  - ceras, [35](#)
- ndim
  - ceras::tensor< T, Allocator >, [111](#)
- nearbyint
  - operation.hpp, [162](#)
- negative
  - ceras, [36](#)
- negative\_relu
  - ceras, [36](#)
- nesterov\_
  - ceras::sgd< Loss, T >, [105](#)
- norm
  - ceras, [36](#)
- normalization\_batch
  - operation.hpp, [162](#)
- ones
  - ceras, [36](#)
- ones\_like
  - ceras, [37](#)
  - operation.hpp, [162](#)
- op\_
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, [118](#)
- op\_type
  - ceras::tensor\_deduction< L, R >, [116](#)
- operation.hpp
  - abs, [152](#)
  - acos, [152](#)
  - acosh, [153](#)
  - asin, [153](#)
  - asinh, [153](#)
  - atan, [153](#)
  - atan2, [154](#)
  - atanh, [154](#)
  - average\_pooling\_2d, [154](#)
  - batch\_normalization, [154](#)
  - cbrt, [154](#)
  - ceil, [155](#)
  - clip, [155](#)
  - concat, [155](#)
  - concatenate, [155](#), [156](#)
  - conv2d, [156](#)
  - cos, [156](#)
  - cosh, [156](#)
  - drop\_out, [156](#)
  - equal, [157](#)
  - erf, [157](#)
  - erfc, [157](#)
  - exp, [158](#)
  - exp2, [158](#)
  - expm1, [158](#)
  - fabs, [158](#)
  - flatten, [159](#)
  - floor, [159](#)
  - hypot, [159](#)
  - identity, [159](#)

- img2col, 159
- llrint, 160
- llround, 160
- log, 160
- log10, 160
- log1p, 161
- log2, 161
- lrint, 161
- lround, 161
- max\_pooling\_2d, 162
- maximum, 162
- nearbyint, 162
- normalization\_batch, 162
- ones\_like, 162
- random\_normal\_like, 163
- reduce\_max, 163
- reduce\_min, 164
- reduce\_sum, 164
- repeat, 164
- reshape, 165
- rint, 165
- round, 165
- sign, 165
- sin, 166
- sinh, 166
- sqr, 168
- sqrt, 166
- tan, 166
- tanh, 167
- transpose, 167
- trunc, 167
- up\_sampling\_2d, 167
- y, 168
- zero\_padding\_2d, 167
- zeros\_like, 168
- Operator
  - ceras, 58
- operator!=
  - ceras, 37
- operator<
  - ceras, 42
- operator<<
  - ceras, 42
- operator<=
  - ceras, 42
- operator>
  - ceras, 43
- operator>=
  - ceras, 43
- operator\*
  - ceras, 37, 38
- operator\*=
  - ceras::tensor< T, Allocator >, 112
- operator()
  - ceras::compiled\_model< Model, Optimizer, Loss >, 75
  - ceras::model< Ex, Ph >, 89
- operator+
  - ceras, 38–40
- operator+=
  - ceras::tensor< T, Allocator >, 112
- operator-
  - ceras, 40–42
  - ceras::tensor< T, Allocator >, 112
- operator-=
  - ceras::tensor< T, Allocator >, 112, 113
- operator/
  - ceras, 42
- operator/=
  - ceras::tensor< T, Allocator >, 113
- operator=
  - ceras::ceras\_private::session< Tsr >, 101
  - ceras::place\_holder< Tsr >, 93, 94
  - ceras::tensor< T, Allocator >, 113
  - ceras::value< T >, 120, 121
  - ceras::variable< Tsr >, 124
- operator==
  - ceras, 43
- operator[]
  - ceras::tensor< T, Allocator >, 113, 114
  - ceras::view\_2d< T >, 131
  - ceras::view\_3d< T >, 134
  - ceras::view\_4d< T >, 137
- optimizer\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, 77
- optimizer\_type
  - ceras::compiled\_model< Model, Optimizer, Loss >, 77
- output
  - ceras::model< Ex, Ph >, 90
- output\_data\_
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 72
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 118
- output\_layer\_type
  - ceras::model< Ex, Ph >, 88
- Place\_Holder
  - ceras, 59
- place\_holder
  - ceras::place\_holder< Tsr >, 92, 93
- place\_holder\_
  - ceras::model< Ex, Ph >, 91
- place\_holder\_type
  - ceras::ceras\_private::session< Tsr >, 99
- place\_holders\_
  - ceras::ceras\_private::session< Tsr >, 103
- plus
  - ceras, 43
- polar
  - ceras, 43
- predict
  - ceras::compiled\_model< Model, Optimizer, Loss >, 75
  - ceras::model< Ex, Ph >, 90

randn  
     ceras, 44  
 randn\_like  
     ceras, 44  
 random  
     ceras, 44  
 random\_generator  
     ceras, 59  
 random\_like  
     ceras, 44  
 random\_normal\_like  
     operation.hpp, 163  
 random\_seed  
     ceras, 59  
 read\_tensor  
     ceras, 45  
 real  
     ceras, 45  
 real\_  
     ceras::complex< Real\_Ex, Imag\_Ex >, 77  
 rebind  
     ceras::ceras\_private::session< Tzor >, 101  
 reduce  
     ceras, 45  
 reduce\_max  
     operation.hpp, 163  
 reduce\_mean  
     ceras, 45  
 reduce\_min  
     operation.hpp, 164  
 reduce\_sum  
     ceras, 46  
     operation.hpp, 164  
 regularizer  
     ceras::regularizer< Float >, 95  
 regularizer\_  
     ceras::variable< Tzor >, 125  
 ReLU  
     ceras, 46  
 relu  
     ceras, 46  
 relu6  
     ceras, 46  
 remember  
     ceras::ceras\_private::session< Tzor >, 101  
 repeat  
     ceras, 46  
     operation.hpp, 164  
 replace\_placeholder\_with\_expression  
     ceras, 47  
 repmat  
     ceras, 47  
 reset  
     ceras::place\_holder< Tzor >, 94  
     ceras::tensor< T, Allocator >, 114  
     ceras::variable< Tzor >, 125  
 Reshape  
     ceras, 47  
 reshape  
     ceras, 47  
     ceras::tensor< T, Allocator >, 114  
     operation.hpp, 165  
 resize  
     ceras::tensor< T, Allocator >, 114  
 restore  
     ceras::ceras\_private::session< Tzor >, 102  
 rho\_  
     ceras::adadelta< Loss, T >, 65  
     ceras::rmsprop< Loss, T >, 98  
 rhs\_input\_data\_  
     ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator,  
         Forward\_Action, Backward\_Action >, 72  
 rhs\_op\_  
     ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator,  
         Forward\_Action, Backward\_Action >, 72  
 rint  
     operation.hpp, 165  
 rms\_prop  
     ceras, 19  
 RMSprop  
     ceras, 59  
 rmsprop  
     ceras::rmsprop< Loss, T >, 97  
 round  
     operation.hpp, 165  
 row  
     ceras::view\_2d< T >, 131  
 row\_  
     ceras::view\_2d< T >, 132  
     ceras::view\_3d< T >, 134  
     ceras::view\_4d< T >, 138  
 row\_begin  
     ceras::view\_2d< T >, 131  
 row\_end  
     ceras::view\_2d< T >, 131, 132  
 row\_type  
     ceras::view\_2d< T >, 128  
 run  
     ceras::ceras\_private::session< Tzor >, 102  
 save  
     ceras::ceras\_private::session< Tzor >, 102  
 save\_tensor  
     ceras, 48  
 save\_weights  
     ceras::model< Ex, Ph >, 90  
 self\_type  
     ceras::tensor< T, Allocator >, 107  
 selu  
     ceras, 48  
 serialize  
     ceras::ceras\_private::session< Tzor >, 102  
 session  
     ceras::ceras\_private::session< Tzor >, 100  
 SGD  
     ceras, 59  
 sgd

- ceras::sgd< Loss, T >, 104
- shape
  - ceras::constant< T, Allocator >, 79
  - ceras::tensor< T, Allocator >, 114
  - ceras::variable< T, Allocator >, 125
  - ceras::view\_2d< T >, 132
- shape\_
  - ceras::tensor< T, Allocator >, 115
- shape\_hint\_
  - ceras::place\_holder\_state< T, Allocator >, 94
- shared\_vector
  - ceras::tensor< T, Allocator >, 107
- shrink\_to
  - ceras::tensor< T, Allocator >, 115
- sigmoid
  - ceras, 48
- sign
  - operation.hpp, 165
- silu
  - ceras, 48
- sin
  - operation.hpp, 166
- sinh
  - operation.hpp, 166
- size
  - ceras::tensor< T, Allocator >, 115
  - ceras::view\_2d< T >, 132
- slice
  - ceras::tensor< T, Allocator >, 115
- Softmax
  - ceras, 48
- softmax
  - ceras, 48, 49
- softplus
  - ceras, 49
- softsign
  - ceras, 49
- sqr
  - operation.hpp, 168
- sqrt
  - operation.hpp, 166
- square
  - ceras, 49
- squared\_loss
  - ceras, 50
- squeeze
  - ceras, 50
- standard\_deviation
  - ceras, 50
- state\_
  - ceras::variable< T, Allocator >, 125
- STB\_IMAGE\_IMPLEMENTATION
  - includes.hpp, 144
- STB\_IMAGE\_RESIZE\_IMPLEMENTATION
  - includes.hpp, 144
- STB\_IMAGE\_WRITE\_IMPLEMENTATION
  - includes.hpp, 144
- std
  - ceras, 50
- Subtract
  - ceras, 50
- sum
  - ceras, 50, 51
- sum\_reduce
  - ceras, 51
- summary
  - ceras::model< Ex, Ph >, 90
- swish
  - ceras, 51
- synchronized\_
  - ceras::regularizer< Float >, 96
- tan
  - operation.hpp, 166
- tanh
  - operation.hpp, 167
- tap
  - ceras::ceras\_private::session< T, Allocator >, 102
- Tensor
  - ceras, 59
- tensor
  - ceras::tensor< T, Allocator >, 108, 109
- tensor\_type
  - ceras::adadelta< Loss, T >, 63
  - ceras::adagrad< Loss, T >, 65
  - ceras::adam< Loss, T >, 68
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 70
  - ceras::gradient\_descent< Loss, T >, 80
  - ceras::place\_holder< T, Allocator >, 92
  - ceras::rmsprop< Loss, T >, 97
  - ceras::sgd< Loss, T >, 104
  - ceras::tensor\_deduction< L, R >, 116
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 118
  - ceras::variable< T, Allocator >, 122
- tesseract
  - ceras, 19
- train\_on\_batch
  - ceras::compiled\_model< Model, Optimizer, Loss >, 75
- trainable
  - ceras::compiled\_model< Model, Optimizer, Loss >, 76
  - ceras::model< Ex, Ph >, 91
  - ceras::variable< T, Allocator >, 125
- trainable\_
  - ceras::variable< T, Allocator >, 126
- transpose
  - operation.hpp, 167
- transposed\_
  - ceras::view\_2d< T >, 133
- trunc
  - operation.hpp, 167
- truncated\_normal
  - ceras, 51



- Unary\_Operator
  - ceras, [60](#)
- unary\_operator
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, [117](#)
- up\_sampling\_2d
  - operation.hpp, [167](#)
- update\_cuda\_gemm\_threshold
  - ceras, [52](#)
- UpSampling2D
  - ceras, [52](#)
- Value
  - ceras, [60](#)
- value
  - ceras::value< T >, [119](#), [120](#)
- value\_type
  - ceras::regularizer< Float >, [95](#)
  - ceras::tensor< T, Allocator >, [107](#)
  - ceras::value< T >, [119](#)
  - ceras::variable< Tsor >, [122](#)
  - ceras::view\_2d< T >, [128](#)
- var
  - ceras, [52](#)
- Variable
  - ceras, [60](#)
- variable
  - ceras::variable< Tsor >, [122](#), [123](#)
- variable\_state\_type
  - ceras::ceras\_private::session< Tsor >, [99](#)
- variable\_type
  - ceras::ceras\_private::session< Tsor >, [100](#)
- variables\_
  - ceras::ceras\_private::session< Tsor >, [103](#)
- variance
  - ceras, [52](#)
- vector\_
  - ceras::tensor< T, Allocator >, [115](#)
- vector\_type
  - ceras::tensor< T, Allocator >, [108](#)
- view\_2d
  - ceras::view\_2d< T >, [128](#), [129](#)
- view\_3d
  - ceras::view\_3d< T >, [133](#)
- view\_4d
  - ceras::view\_4d< T >, [135](#)
- write\_tensor
  - ceras, [52](#)
- y
  - operation.hpp, [168](#)
- zero\_padding\_2d
  - operation.hpp, [167](#)
- zeros
  - ceras, [52](#)
- zeros\_like
  - ceras, [53](#)
  - operation.hpp, [168](#)