

ceras

Generated by Doxygen 1.9.1

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
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	18
5.1.1.4 default_allocator	18
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]	19
5.1.2.3 abs_loss()	20
5.1.2.4 Add()	20
5.1.2.5 add()	20
5.1.2.6 amax()	20
5.1.2.7 amin()	20
5.1.2.8 arg()	20
5.1.2.9 as_tensor()	21
5.1.2.10 AveragePooling2D()	21
5.1.2.11 BatchNormalization() [1/2]	21
5.1.2.12 BatchNormalization() [2/2]	21
5.1.2.13 binary_cross_entropy_loss()	22
5.1.2.14 clip()	22
5.1.2.15 computation_graph()	22
5.1.2.16 concatenate()	23
5.1.2.17 Concatenate()	23
5.1.2.18 conj()	23
5.1.2.19 Conv2D()	24
5.1.2.20 copy()	24
5.1.2.21 cross_entropy()	25

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

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

5.1.2.106 operator<()	41
5.1.2.107 operator<=()	41
5.1.2.108 operator==() [1/2]	41
5.1.2.109 operator==() [2/2]	42
5.1.2.110 operator>()	42
5.1.2.111 operator>=()	42
5.1.2.112 plus()	42
5.1.2.113 polar()	42
5.1.2.114 randn()	43
5.1.2.115 randn_like()	43
5.1.2.116 random()	43
5.1.2.117 random_like()	43
5.1.2.118 read_tensor()	43
5.1.2.119 real()	44
5.1.2.120 reduce()	44
5.1.2.121 reduce_mean() [1/2]	44
5.1.2.122 reduce_mean() [2/2]	44
5.1.2.123 reduce_sum() [1/2]	44
5.1.2.124 reduce_sum() [2/2]	45
5.1.2.125 relu()	45
5.1.2.126 ReLU()	45
5.1.2.127 relu6()	45
5.1.2.128 repeat()	45
5.1.2.129 replace_placeholder_with_expression()	45
5.1.2.130 repmat()	46
5.1.2.131 Reshape()	46
5.1.2.132 reshape()	46
5.1.2.133 save_tensor()	46
5.1.2.134 selu()	47
5.1.2.135 sigmoid()	47
5.1.2.136 silu()	47
5.1.2.137 Softmax()	47
5.1.2.138 softmax() [1/2]	47
5.1.2.139 softmax() [2/2]	47
5.1.2.140 softplus()	48
5.1.2.141 softsign()	48
5.1.2.142 square()	48
5.1.2.143 squared_loss()	48
5.1.2.144 squeeze()	49
5.1.2.145 standard_deviation()	49
5.1.2.146 std()	49
5.1.2.147 Subtract()	49

5.1.2.148 <code>sum()</code> [1/2]	49
5.1.2.149 <code>sum()</code> [2/2]	50
5.1.2.150 <code>sum_reduce()</code>	50
5.1.2.151 <code>swish()</code>	50
5.1.2.152 <code>truncated_normal()</code>	50
5.1.2.153 <code>update_cuda_gemm_threshold()</code>	50
5.1.2.154 <code>UpSampling2D()</code>	50
5.1.2.155 <code>var()</code>	51
5.1.2.156 <code>variance()</code>	51
5.1.2.157 <code>write_tensor()</code>	51
5.1.2.158 <code>zeros()</code>	51
5.1.2.159 <code>zeros_like()</code>	51
5.1.3 Variable Documentation	51
5.1.3.1 <code>Adadelta</code>	52
5.1.3.2 <code>Adagrad</code>	52
5.1.3.3 <code>Adam</code>	52
5.1.3.4 <code>Binary_Operator</code>	52
5.1.3.5 <code>BinaryCrossentropy</code>	53
5.1.3.6 <code>BinaryCrossEntropy</code>	53
5.1.3.7 <code>CategoricalCrossentropy</code>	53
5.1.3.8 <code>CategoricalCrossEntropy</code>	53
5.1.3.9 <code>Complex</code>	54
5.1.3.10 <code>Constant</code>	54
5.1.3.11 <code>Expression</code>	54
5.1.3.12 <code>Hinge</code>	54
5.1.3.13 <code>is_binary_operator_v</code>	54
5.1.3.14 <code>is_complex_v</code>	55
5.1.3.15 <code>is_constant_v</code>	55
5.1.3.16 <code>is_place_holder_v</code>	55
5.1.3.17 <code>is_tensor_v</code>	55
5.1.3.18 <code>is_unary_operator_v</code>	55
5.1.3.19 <code>is_value_v</code>	55
5.1.3.20 <code>is_variable_v</code>	55
5.1.3.21 <code>MAE</code>	56
5.1.3.22 <code>make_binary_operator</code>	56
5.1.3.23 <code>make_unary_operator</code>	56
5.1.3.24 <code>MeanAbsoluteError</code>	56
5.1.3.25 <code>MeanSquaredError</code>	57
5.1.3.26 <code>MSE</code>	57
5.1.3.27 <code>Operator</code>	57
5.1.3.28 <code>Place_Holder</code>	57
5.1.3.29 <code>random_generator</code>	58

5.1.3.30 random_seed	58
5.1.3.31 RMSprop	58
5.1.3.32 SGD	58
5.1.3.33 Tensor	58
5.1.3.34 Unary_Operator	59
5.1.3.35 Value	59
5.1.3.36 Variable	59
5.2 ceras::ceras_private Namespace Reference	59
5.3 ceras::dataset Namespace Reference	59
5.4 ceras::dataset::fashion_mnist Namespace Reference	59
5.4.1 Function Documentation	59
5.4.1.1 load_data()	59
5.5 ceras::dataset::mnist Namespace Reference	60
5.5.1 Function Documentation	60
5.5.1.1 load_data()	60
6 Class Documentation	63
6.1 ceras::adadelta< 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 adadelta()	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, Rhs_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 Constructor & Destructor Documentation	78
6.7.1.1 constant()	78
6.7.2 Member Function Documentation	78
6.7.2.1 backward()	79
6.7.2.2 forward()	79
6.7.2.3 shape()	79
6.7.3 Member Data Documentation	79
6.7.3.1 data_	79
6.8 ceras::gradient_descent< Loss, T > Struct Template Reference	79
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	80
6.8.3.1 forward()	80
6.8.4 Member Data Documentation	80
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 ceras::is_constant< T > Struct Template Reference	83
6.14 ceras::is_constant< constant< Tsor > > Struct Template Reference	83
6.15 ceras::is_place_holder< T > Struct Template Reference	83
6.16 ceras::is_place_holder< place_holder< Tsor > > Struct Template Reference	84
6.17 ceras::is_tensor< T > Struct Template Reference	84

6.18 ceras::is_tensor< tensor< T, A > > Struct Template Reference	84
6.19 ceras::is_unary_operator< T > Struct Template Reference	85
6.20 ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > > Struct Template Reference	85
6.21 ceras::is_value< T > Struct Template Reference	85
6.22 ceras::is_value< value< T > > Struct Template Reference	86
6.23 ceras::is_variable< T > Struct Template Reference	86
6.24 ceras::is_variable< variable< Tsor > > Struct Template Reference	86
6.25 ceras::model< Ex, Ph > Struct Template Reference	87
6.25.1 Detailed Description	87
6.25.2 Member Typedef Documentation	88
6.25.2.1 input_layer_type	88
6.25.2.2 output_layer_type	88
6.25.3 Constructor & Destructor Documentation	88
6.25.3.1 model() [1/2]	88
6.25.3.2 model() [2/2]	89
6.25.4 Member Function Documentation	89
6.25.4.1 compile()	89
6.25.4.2 input() [1/2]	89
6.25.4.3 input() [2/2]	89
6.25.4.4 load_weights()	90
6.25.4.5 operator>()	90
6.25.4.6 output() [1/2]	90
6.25.4.7 output() [2/2]	90
6.25.4.8 predict() [1/3]	91
6.25.4.9 predict() [2/3]	91
6.25.4.10 predict() [3/3]	92
6.25.4.11 save_weights()	92
6.25.4.12 summary()	92
6.25.4.13 trainable()	92
6.25.5 Member Data Documentation	92
6.25.5.1 expression_	92
6.25.5.2 input_layer_	93
6.25.5.3 output_layer_	93
6.25.5.4 place_holder_	93
6.26 ceras::place_holder< Tsor > Struct Template Reference	93
6.26.1 Member Typedef Documentation	94
6.26.1.1 tensor_type	94
6.26.2 Constructor & Destructor Documentation	94
6.26.2.1 place_holder() [1/4]	94
6.26.2.2 place_holder() [2/4]	94
6.26.2.3 place_holder() [3/4]	94

6.26.2.4 place_holder() [4 / 4]	94
6.26.3 Member Function Documentation	94
6.26.3.1 backward()	95
6.26.3.2 bind()	95
6.26.3.3 forward()	95
6.26.3.4 operator=() [1 / 2]	95
6.26.3.5 operator=() [2 / 2]	95
6.26.3.6 reset()	95
6.27 ceras::place_holder_state< Tsr > Struct Template Reference	96
6.27.1 Member Data Documentation	96
6.27.1.1 data_	96
6.27.1.2 shape_hint_	96
6.28 ceras::regularizer< Float > Struct Template Reference	96
6.28.1 Member Typedef Documentation	97
6.28.1.1 value_type	97
6.28.2 Constructor & Destructor Documentation	97
6.28.2.1 regularizer()	97
6.28.3 Member Data Documentation	97
6.28.3.1 l1_	97
6.28.3.2 l2_	97
6.28.3.3 synchronized_	98
6.29 ceras::rmsprop< Loss, T > Struct Template Reference	98
6.29.1 Member Typedef Documentation	98
6.29.1.1 tensor_type	98
6.29.2 Constructor & Destructor Documentation	99
6.29.2.1 rmsprop()	99
6.29.3 Member Function Documentation	99
6.29.3.1 forward()	99
6.29.4 Member Data Documentation	99
6.29.4.1 decay_	99
6.29.4.2 iterations_	99
6.29.4.3 learning_rate_	99
6.29.4.4 loss_	100
6.29.4.5 rho_	100
6.30 ceras::ceras_private::session< Tsr > Struct Template Reference	100
6.30.1 Member Typedef Documentation	101
6.30.1.1 place_holder_type	101
6.30.1.2 variable_state_type	101
6.30.1.3 variable_type	101
6.30.2 Constructor & Destructor Documentation	101
6.30.2.1 session() [1 / 3]	101
6.30.2.2 session() [2 / 3]	101

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

6.32.2.4 <code>tensor()</code> [4/7]	109
6.32.2.5 <code>tensor()</code> [5/7]	110
6.32.2.6 <code>tensor()</code> [6/7]	110
6.32.2.7 <code>tensor()</code> [7/7]	110
6.32.3 Member Function Documentation	110
6.32.3.1 <code>as_scalar()</code>	110
6.32.3.2 <code>as_type()</code>	110
6.32.3.3 <code>begin()</code> [1/2]	110
6.32.3.4 <code>begin()</code> [2/2]	111
6.32.3.5 <code>cbegin()</code>	111
6.32.3.6 <code>cend()</code>	111
6.32.3.7 <code>copy()</code>	111
6.32.3.8 <code>creep_to()</code>	111
6.32.3.9 <code>data()</code> [1/2]	111
6.32.3.10 <code>data()</code> [2/2]	111
6.32.3.11 <code>deep_copy()</code> [1/2]	112
6.32.3.12 <code>deep_copy()</code> [2/2]	112
6.32.3.13 <code>empty()</code>	112
6.32.3.14 <code>end()</code> [1/2]	112
6.32.3.15 <code>end()</code> [2/2]	112
6.32.3.16 <code>map()</code>	112
6.32.3.17 <code>ndim()</code>	113
6.32.3.18 <code>operator*=()</code> [1/2]	113
6.32.3.19 <code>operator*=()</code> [2/2]	113
6.32.3.20 <code>operator+=()</code> [1/2]	113
6.32.3.21 <code>operator+=()</code> [2/2]	113
6.32.3.22 <code>operator-()</code>	113
6.32.3.23 <code>operator-=()</code> [1/2]	114
6.32.3.24 <code>operator-=()</code> [2/2]	114
6.32.3.25 <code>operator/=()</code> [1/2]	114
6.32.3.26 <code>operator/=()</code> [2/2]	114
6.32.3.27 <code>operator=()</code> [1/2]	114
6.32.3.28 <code>operator=()</code> [2/2]	114
6.32.3.29 <code>operator[]()</code> [1/2]	115
6.32.3.30 <code>operator[]()</code> [2/2]	115
6.32.3.31 <code>reset()</code>	115
6.32.3.32 <code>reshape()</code>	115
6.32.3.33 <code>resize()</code>	115
6.32.3.34 <code>shape()</code>	116
6.32.3.35 <code>shrink_to()</code>	116
6.32.3.36 <code>size()</code>	116
6.32.3.37 <code>slice()</code>	116

6.32.4 Member Data Documentation	116
6.32.4.1 memory_offset_	116
6.32.4.2 shape_	116
6.32.4.3 vector_	117
6.33 ceras::tensor_deduction< L, R > Struct Template Reference	117
6.33.1 Member Typedef Documentation	117
6.33.1.1 op_type	117
6.33.1.2 tensor_type	117
6.34 ceras::unary_operator< Operator, Forward_Action, Backward_Action > Struct Template Reference	117
6.34.1 Constructor & Destructor Documentation	118
6.34.1.1 unary_operator()	118
6.34.2 Member Function Documentation	118
6.34.2.1 backward()	118
6.34.2.2 forward()	118
6.34.3 Member Data Documentation	119
6.34.3.1 backward_action_	119
6.34.3.2 forward_action_	119
6.34.3.3 input_data_	119
6.34.3.4 op_	119
6.34.3.5 output_data_	119
6.34.3.6 tensor_type	119
6.35 ceras::value< T > Struct Template Reference	120
6.35.1 Member Typedef Documentation	120
6.35.1.1 value_type	120
6.35.2 Constructor & Destructor Documentation	120
6.35.2.1 value() [1/4]	121
6.35.2.2 value() [2/4]	121
6.35.2.3 value() [3/4]	121
6.35.2.4 value() [4/4]	121
6.35.3 Member Function Documentation	121
6.35.3.1 backward()	121
6.35.3.2 forward()	121
6.35.3.3 operator=() [1/2]	122
6.35.3.4 operator=() [2/2]	122
6.35.4 Member Data Documentation	122
6.35.4.1 data_	122
6.36 ceras::variable< Tsor > Struct Template Reference	122
6.36.1 Member Typedef Documentation	123
6.36.1.1 tensor_type	123
6.36.1.2 value_type	123
6.36.2 Constructor & Destructor Documentation	123
6.36.2.1 variable() [1/4]	124

6.36.2.2 variable() [2/4]	124
6.36.2.3 variable() [3/4]	124
6.36.2.4 variable() [4/4]	124
6.36.3 Member Function Documentation	124
6.36.3.1 backward()	124
6.36.3.2 contexts() [1/2]	124
6.36.3.3 contexts() [2/2]	125
6.36.3.4 data() [1/2]	125
6.36.3.5 data() [2/2]	125
6.36.3.6 forward()	125
6.36.3.7 gradient() [1/2]	125
6.36.3.8 gradient() [2/2]	125
6.36.3.9 operator=() [1/2]	125
6.36.3.10 operator=() [2/2]	126
6.36.3.11 reset()	126
6.36.3.12 shape()	126
6.36.3.13 trainable() [1/2]	126
6.36.3.14 trainable() [2/2]	126
6.36.4 Member Data Documentation	126
6.36.4.1 regularizer_	126
6.36.4.2 state_	127
6.36.4.3 trainable_	127
6.37 ceras::variable_state< Tsor > Struct Template Reference	127
6.37.1 Member Data Documentation	127
6.37.1.1 contexts_	127
6.37.1.2 data_	127
6.37.1.3 gradient_	128
6.38 ceras::view_2d< T > Struct Template Reference	128
6.38.1 Member Typedef Documentation	129
6.38.1.1 col_type	129
6.38.1.2 const_col_type	129
6.38.1.3 const_row_type	129
6.38.1.4 row_type	129
6.38.1.5 value_type	129
6.38.2 Constructor & Destructor Documentation	129
6.38.2.1 view_2d() [1/3]	130
6.38.2.2 view_2d() [2/3]	130
6.38.2.3 view_2d() [3/3]	130
6.38.3 Member Function Documentation	130
6.38.3.1 begin()	130
6.38.3.2 col()	130
6.38.3.3 col_begin() [1/2]	131

6.38.3.4 col_begin() [2/2]	131
6.38.3.5 col_end() [1/2]	131
6.38.3.6 col_end() [2/2]	131
6.38.3.7 data() [1/2]	131
6.38.3.8 data() [2/2]	131
6.38.3.9 end()	132
6.38.3.10 operator[]() [1/2]	132
6.38.3.11 operator[]() [2/2]	132
6.38.3.12 row()	132
6.38.3.13 row_begin() [1/2]	132
6.38.3.14 row_begin() [2/2]	132
6.38.3.15 row_end() [1/2]	133
6.38.3.16 row_end() [2/2]	133
6.38.3.17 shape()	133
6.38.3.18 size()	133
6.38.4 Member Data Documentation	133
6.38.4.1 col_	133
6.38.4.2 data_	133
6.38.4.3 row_	134
6.38.4.4 transposed_	134
6.39 ceras::view_3d< T > Struct Template Reference	134
6.39.1 Constructor & Destructor Documentation	134
6.39.1.1 view_3d()	134
6.39.2 Member Function Documentation	135
6.39.2.1 operator[]() [1/2]	135
6.39.2.2 operator[]() [2/2]	135
6.39.3 Member Data Documentation	135
6.39.3.1 channel_	135
6.39.3.2 col_	135
6.39.3.3 data_	135
6.39.3.4 row_	136
6.40 ceras::view_4d< T > Struct Template Reference	136
6.40.1 Detailed Description	136
6.40.2 Constructor & Destructor Documentation	136
6.40.2.1 view_4d()	137
6.40.3 Member Function Documentation	138
6.40.3.1 operator[]() [1/2]	138
6.40.3.2 operator[]() [2/2]	138
6.40.4 Member Data Documentation	139
6.40.4.1 batch_size_	139
6.40.4.2 channel_	139
6.40.4.3 col_	139

6.40.4.4 data_	139
6.40.4.5 row_	139
7 File Documentation	141
7.1 /data/structured_folders/workspace/github.repo/ceras/include/activation.hpp File Reference	141
7.2 /data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp File Reference	142
7.3 /data/structured_folders/workspace/github.repo/ceras/include/complex_operator.hpp File Reference	142
7.4 /data/structured_folders/workspace/github.repo/ceras/include/config.hpp File Reference	144
7.5 /data/structured_folders/workspace/github.repo/ceras/include/constant.hpp File Reference	144
7.6 /data/structured_folders/workspace/github.repo/ceras/include/dataset.hpp File Reference	145
7.7 /data/structured_folders/workspace/github.repo/ceras/include/includes.hpp File Reference	145
7.7.1 Macro Definition Documentation	146
7.7.1.1 STB_IMAGE_IMPLEMENTATION	146
7.7.1.2 STB_IMAGE_RESIZE_IMPLEMENTATION	146
7.7.1.3 STB_IMAGE_WRITE_IMPLEMENTATION	146
7.8 /data/structured_folders/workspace/github.repo/ceras/include/layer.hpp File Reference	147
7.9 /data/structured_folders/workspace/github.repo/ceras/include/loss.hpp File Reference	148
7.10 /data/structured_folders/workspace/github.repo/ceras/include/model.hpp File Reference	149
7.11 /data/structured_folders/workspace/github.repo/ceras/include/operation.hpp File Reference	150
7.11.1 Function Documentation	154
7.11.1.1 abs()	155
7.11.1.2 acos()	155
7.11.1.3 acosh()	155
7.11.1.4 asin()	155
7.11.1.5 asinh()	156
7.11.1.6 atan()	156
7.11.1.7 atan2()	156
7.11.1.8 atanh()	156
7.11.1.9 average_pooling_2d()	157
7.11.1.10 batch_normalization()	157
7.11.1.11 cbrt()	157
7.11.1.12 ceil()	157
7.11.1.13 clip()	157
7.11.1.14 concat() [1/2]	158
7.11.1.15 concat() [2/2]	158
7.11.1.16 concatenate() [1/2]	158
7.11.1.17 concatenate() [2/2]	158
7.11.1.18 conv2d()	158
7.11.1.19 cos()	159
7.11.1.20 cosh()	159
7.11.1.21 drop_out()	159
7.11.1.22 equal()	159

7.11.1.23 erf()	160
7.11.1.24 erfc()	160
7.11.1.25 exp()	160
7.11.1.26 exp2()	161
7.11.1.27 expm1()	161
7.11.1.28 fabs()	161
7.11.1.29 flatten()	161
7.11.1.30 floor()	162
7.11.1.31 hypot()	162
7.11.1.32 identity()	162
7.11.1.33 img2col()	162
7.11.1.34 llrint()	162
7.11.1.35 llround()	163
7.11.1.36 log()	163
7.11.1.37 log10()	163
7.11.1.38 log1p()	163
7.11.1.39 log2()	164
7.11.1.40 lrint()	164
7.11.1.41 lround()	164
7.11.1.42 max_pooling_2d()	164
7.11.1.43 maximum()	164
7.11.1.44 nearbyint()	165
7.11.1.45 normalization_batch()	165
7.11.1.46 ones_like()	165
7.11.1.47 random_normal_like()	165
7.11.1.48 reduce_max()	166
7.11.1.49 reduce_min()	166
7.11.1.50 reduce_sum()	166
7.11.1.51 repeat()	167
7.11.1.52 reshape()	167
7.11.1.53 rint()	167
7.11.1.54 round()	168
7.11.1.55 sign()	168
7.11.1.56 sin()	168
7.11.1.57 sinh()	169
7.11.1.58 sqrt()	169
7.11.1.59 tan()	169
7.11.1.60 tanh()	169
7.11.1.61 transpose()	170
7.11.1.62 trunc()	170
7.11.1.63 up_sampling_2d()	170
7.11.1.64 zero_padding_2d()	170

7.11.1.65 zeros_like()	171
7.11.2 Variable Documentation	171
7.11.2.1 sqr	171
7.11.2.2 y	171
7.12 /data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp File Reference	171
7.13 /data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp File Reference	172
7.14 /data/structured_folders/workspace/github.repo/ceras/include/session.hpp File Reference	173
7.15 /data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp File Reference	174
7.16 /data/structured_folders/workspace/github.repo/ceras/include/value.hpp File Reference	177
7.17 /data/structured_folders/workspace/github.repo/ceras/include/variable.hpp File Reference	178
7.18 /data/structured_folders/workspace/github.repo/ceras/include/xmodel.hpp File Reference	179
Index	181

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

ceras	9
ceras::ceras_private	59
ceras::dataset	59
ceras::dataset::fashion_mnist	59
ceras::dataset::mnist	60

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 >	79
ceras::place_holder< Tsor >	93
ceras::rmsprop< Loss, T >	98
ceras::sgd< Loss, T >	104
ceras::tensor< T, Allocator >	107
ceras::unary_operator< Operator, Forward_Action, Backward_Action >	117
ceras::value< T >	120
ceras::variable< Tsor >	122
enable_shared	
ceras::adadelta< Loss, T >	63
ceras::adagrad< Loss, T >	65
ceras::adam< Loss, T >	67
ceras::gradient_descent< Loss, T >	79
ceras::rmsprop< Loss, T >	98
ceras::sgd< Loss, T >	104
enable_shared_state	
ceras::place_holder< Tsor >	93
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 >	96
ceras::regularizer< Float >	96
ceras::regularizer< value_type >	96
ceras::ceras_private::session< Tsor >	100
ceras::tensor_deduction< L, R >	117
ceras::tensor_deduction< Lhs_Operator, Rhc_Operator >	117
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 >	127
ceras::view_2d< T >	128
ceras::view_3d< T >	134
ceras::view_4d< T >	136

Chapter 3

Class Index

3.1 Class List

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

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::compiled_model< Model, Optimizer, Loss >	72
ceras::complex< Real_Ex, Imag_Ex >	77
ceras::constant< Tzor >	78
ceras::gradient_descent< Loss, T >	79
ceras::is_binary_operator< T >	81
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > >	82
ceras::is_complex< T >	82
ceras::is_complex< complex< Real_Ex, Imag_Ex > >	82
ceras::is_constant< T >	83
ceras::is_constant< constant< Tzor > >	83
ceras::is_place_holder< T >	83
ceras::is_place_holder< place_holder< Tzor > >	84
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< Tzor > >	86
ceras::model< Ex, Ph >	87
ceras::place_holder< Tzor >	93
ceras::place_holder_state< Tzor >	96
ceras::regularizer< Float >	96
ceras::rmsprop< Loss, T >	98
ceras::ceras_private::session< Tzor >	100
ceras::sgd< Loss, T >	104
ceras::tensor< T, Allocator >	107
ceras::tensor_deduction< L, R >	117
ceras::unary_operator< Operator, Forward_Action, Backward_Action >	117

ceras::value< T >	120
ceras::variable< Tsor >	122
ceras::variable_state< Tsor >	127
ceras::view_2d< T >	128
ceras::view_3d< T >	134
ceras::view_4d< T >	136

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/data/structured_folders/workspace/github.repo/ceras/include/activation.hpp	141
/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp	142
/data/structured_folders/workspace/github.repo/ceras/include/complex_operator.hpp	142
/data/structured_folders/workspace/github.repo/ceras/include/config.hpp	144
/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp	144
/data/structured_folders/workspace/github.repo/ceras/include/dataset.hpp	145
/data/structured_folders/workspace/github.repo/ceras/include/includes.hpp	145
/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp	147
/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp	148
/data/structured_folders/workspace/github.repo/ceras/include/model.hpp	149
/data/structured_folders/workspace/github.repo/ceras/include/operation.hpp	150
/data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp	171
/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp	172
/data/structured_folders/workspace/github.repo/ceras/include/session.hpp	173
/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp	174
/data/structured_folders/workspace/github.repo/ceras/include/value.hpp	177
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp	178
/data/structured_folders/workspace/github.repo/ceras/include/xmodel.hpp	179

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](#)
- 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 [adadelta](#)
- 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. 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>.
- `template<Expression Ex>`
`auto silu (Ex const &ex) noexcept`
An alias name of activation swish.
- `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 Rh_Expression>
constexpr auto [mean_squared_logarithmic_error](#) (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rh_Expression>
constexpr auto [squared_loss](#) (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rh_Expression>
constexpr auto [mean_squared_error](#) (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept

- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto mse (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto abs_loss (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto mean_absolute_error (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto mae (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto cross_entropy (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto binary_cross_entropy_loss (Lhs_Expression const &ground_truth, Rhhs_Expression const &prediction) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto cross_entropy_loss (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto hinge_loss (Lhs_Expression const &lhs_ex, Rhhs_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 Rhhs_Expression>`
`constexpr auto plus (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto operator+ (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`auto operator* (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`
`constexpr auto negative (Ex const &ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto elementwise_product (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto elementwise_multiply (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto hadamard_product (Lhs_Expression const &lhs_ex, Rhhs_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`
- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto minus (Lhs_Expression const &lhs_ex, Rhhs_Expression const &rhs_ex) noexcept`

- `template<Expression Lhs_Expression, Expression Rhhs_Expression>`
`constexpr auto operator- (Lhs_Expression const &lhs_ex, Rhhs_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 endpoint=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

<i>c</i>	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 Rhs_Expression>
constexpr auto ceras::abs_loss (
    Lhs_Expression const & lhs_ex,
    Rhs_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

c	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.
<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 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]
```

5.1.2.22 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]
```

5.1.2.23 deep_copy()

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

5.1.2.24 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 };
```

5.1.2.25 Dropout()

```
template<typename T >
auto ceras::Dropout (
    T factor ) [inline], [noexcept]
```

Applies Dropout to the input.

5.1.2.26 elementwise_divide()

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

5.1.2.27 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]
```

5.1.2.28 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]
```

5.1.2.29 elementwise_product() [2/2]

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

5.1.2.30 elu()

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

5.1.2.31 ELU()

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

Exponential Linear Unit.

5.1.2.32 empty()

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

5.1.2.33 exponential()

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

5.1.2.34 Flatten()

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

Flattens the input. Does not affect the batch size.

5.1.2.35 gelu()

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

5.1.2.36 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 )
```

5.1.2.37 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 )
```

5.1.2.38 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 )
```

5.1.2.39 get_default_session()

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

5.1.2.40 glorot_uniform()

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


5.1.2.41 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]
```

5.1.2.42 hadamard_product() [2/2]

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

5.1.2.43 hard_sigmoid()

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

5.1.2.44 has_inf()

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

5.1.2.45 has_nan()

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

5.1.2.46 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]
```

5.1.2.47 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.
----------	-----------------------

5.1.2.48 Input()

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

5.1.2.49 is_valid()

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

5.1.2.50 leaky_relu()

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

5.1.2.51 LeakyReLU()

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

leaky relu activation function.

5.1.2.52 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]
```

5.1.2.53 load_tensor()

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

5.1.2.54 mae()

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

5.1.2.55 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]
```

5.1.2.56 make_trainable()

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

Setting an expression's trainable flag

5.1.2.57 max() [1/2]

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

5.1.2.58 max() [2/2]

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

5.1.2.59 MaxPooling2D()

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

Max pooling operation for 2D spatial data.

5.1.2.60 mean() [1/2]

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

5.1.2.61 mean() [2/2]

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

5.1.2.62 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]
```

5.1.2.63 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 );
```

5.1.2.64 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]
```

5.1.2.65 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]
```

5.1.2.66 min() [1/2]

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

5.1.2.67 min() [2/2]

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

5.1.2.68 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]
```

5.1.2.69 minus() [2/2]

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

5.1.2.70 mse()

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

5.1.2.71 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 };
```

5.1.2.72 multiply() [1/2]

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

5.1.2.73 multiply() [2/2]

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

5.1.2.74 negative()

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

5.1.2.75 negative_relu()

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

5.1.2.76 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 );
```

5.1.2.77 norm() [2/2]

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

5.1.2.78 ones()

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

5.1.2.79 ones_like()

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

5.1.2.80 operator"!="()

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

5.1.2.81 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.

5.1.2.82 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;
```


5.1.2.83 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.

5.1.2.84 operator*() [4/7]

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

5.1.2.85 operator*() [5/7]

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

5.1.2.86 operator*() [6/7]

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

5.1.2.87 operator*() [7/7]

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

5.1.2.88 operator+() [1/8]

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

Returns the complex expression.

5.1.2.89 operator+() [2/8]

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

Sums up a complex expression and an expression.

5.1.2.90 operator+() [3/8]

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

Sums up two complex expressions.

5.1.2.91 operator+() [4/8]

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

Sums up a complex expression and an expression.

5.1.2.92 operator+() [5/8]

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

5.1.2.93 operator+() [6/8]

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

5.1.2.94 operator+() [7/8]

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

5.1.2.95 operator+() [8/8]

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

5.1.2.96 operator-() [1/8]

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

Negatives the complex expression.

5.1.2.97 operator-() [2/8]

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

Subtracts an expression from a compression expression.

5.1.2.98 operator-() [3/8]

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

Subtracts one complex expression from the other one.

5.1.2.99 operator-() [4/8]

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

Subtracts a complex expression from an expression.

5.1.2.100 operator-() [5/8]

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

5.1.2.101 operator-() [6/8]

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

5.1.2.102 operator-() [7/8]

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

5.1.2.103 operator-() [8/8]

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

5.1.2.104 operator/()

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

5.1.2.105 operator<()

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

5.1.2.106 operator<<()

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

5.1.2.107 operator<=()

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

5.1.2.108 operator==() [1/2]

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

5.1.2.109 operator==() [2/2]

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

5.1.2.110 operator>()

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

5.1.2.111 operator>=()

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

5.1.2.112 plus()

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

5.1.2.113 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 );

```

5.1.2.114 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} )

```

5.1.2.115 randn_like()

```

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

```

5.1.2.116 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} )

```

5.1.2.117 random_like()

```

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

```

5.1.2.118 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 )

```

5.1.2.119 real()

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

@brief Returns the real part of the complex expression.

Parameters

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

5.1.2.120 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]
```

5.1.2.121 reduce_mean() [1/2]

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

5.1.2.122 reduce_mean() [2/2]

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

5.1.2.123 reduce_sum() [1/2]

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


5.1.2.124 `reduce_sum()` [2/2]

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

5.1.2.125 `relu()`

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

5.1.2.126 `ReLU()`

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

Rectified Linear Unit activation function.

5.1.2.127 `relu6()`

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

5.1.2.128 `repeat()`

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

5.1.2.129 `replace_placeholder_with_expression()`

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

Replacing a [place_holder](#) with an expression.

Parameters

<i>ex</i>	Can be a unary operator, binary operator, variable, place_holder , 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*

5.1.2.130 repmat()

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

5.1.2.131 Reshape()

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

Reshapes inputs into the given shape.

5.1.2.132 reshape()

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

5.1.2.133 save_tensor()

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

5.1.2.134 selu()

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

5.1.2.135 sigmoid()

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

5.1.2.136 silu()

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

An alias name of activation swish.

5.1.2.137 Softmax()

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

Softmax activation function.

5.1.2.138 softmax() [1/2]

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

5.1.2.139 softmax() [2/2]

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

5.1.2.140 softplus()

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

5.1.2.141 softsign()

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

5.1.2.142 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);
```

5.1.2.143 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]
```

5.1.2.144 squeeze()

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

5.1.2.145 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]
```

5.1.2.146 std()

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

5.1.2.147 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 };
```

5.1.2.148 sum() [1/2]

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

5.1.2.149 sum() [2/2]

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

5.1.2.150 sum_reduce()

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

5.1.2.151 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>.

5.1.2.152 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} )
```

5.1.2.153 update_cuda_gemm_threshold()

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

5.1.2.154 UpSampling2D()

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

Upsampling layer for 2D inputs.

5.1.2.155 var()

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

5.1.2.156 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]
```

5.1.2.157 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 )
```

5.1.2.158 zeros()

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

5.1.2.159 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( MeanSquareError(), 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, 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, 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.

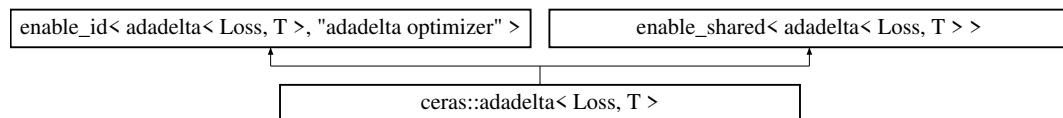
Chapter 6

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_
```

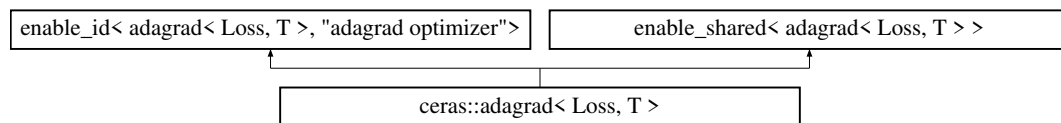
The documentation for this struct was generated from the following file:

- /data/structured_folders/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_
```

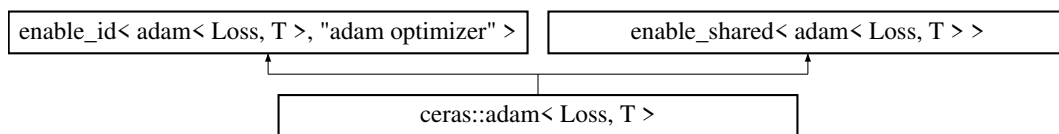
The documentation for this struct was generated from the following file:

- /data/structured_folders/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_
```

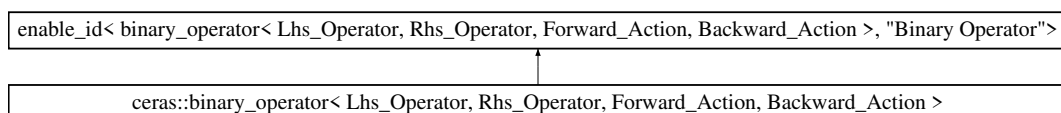
The documentation for this struct was generated from the following file:

- /data/structured_folders/workspace/github.repo/ceras/include/[optimizer.hpp](#)

6.4 ceras::binary_operator< Lhs_Operator, Rhb_Operator, Forward_Action, Backward_Action > Struct Template Reference

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

Inheritance diagram for ceras::binary_operator< Lhs_Operator, Rhb_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:

- [/data/structured_folders/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:

- /data/structured_folders/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_
```

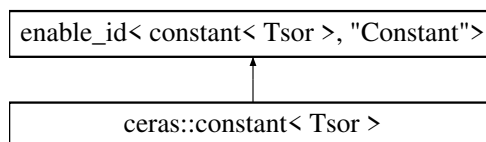
The documentation for this struct was generated from the following file:

- /data/structured_folders/workspace/github.repo/ceras/include/complex_operator.hpp

6.7 ceras::constant< Tsor > Struct Template Reference

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

Inheritance diagram for ceras::constant< Tsor >:



Public Member Functions

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

Public Attributes

- Tsor [data_](#)

6.7.1 Constructor & Destructor Documentation

6.7.1.1 constant()

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

6.7.2 Member Function Documentation

6.7.2.1 backward()

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

6.7.2.2 forward()

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

6.7.2.3 shape()

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

6.7.3 Member Data Documentation

6.7.3.1 data_

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

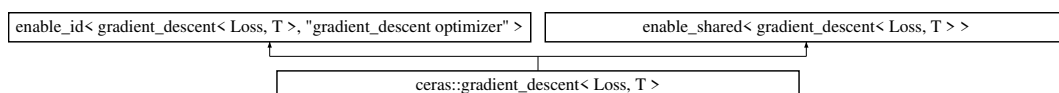
The documentation for this struct was generated from the following file:

- /data/structured_folders/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_
```

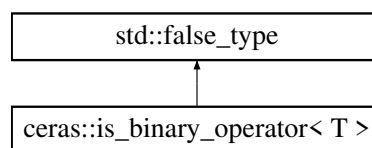
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 >:



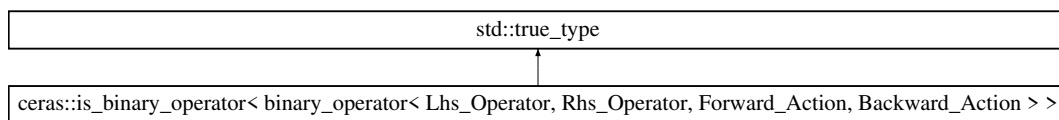
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 > >`:



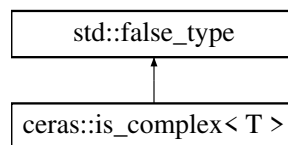
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 >`:



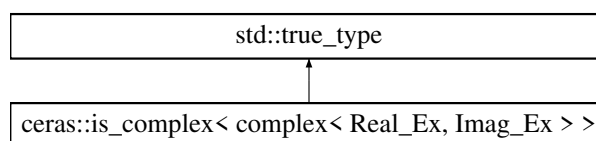
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 > >`:



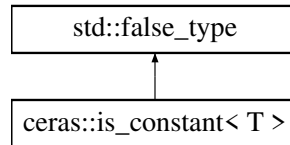
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 >:



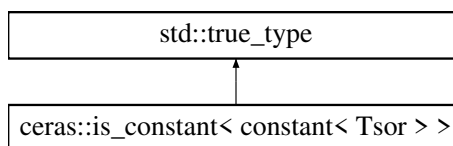
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 > >:



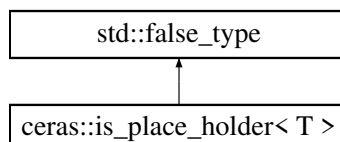
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 >:



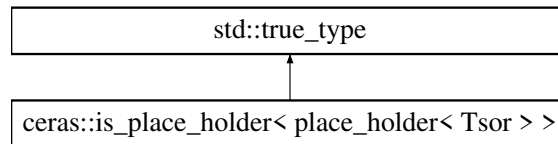
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 > >`:



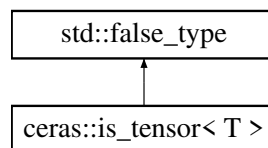
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 >`:



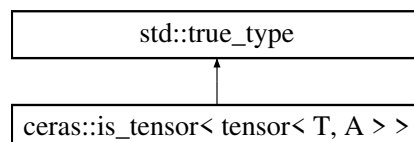
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 > >`:



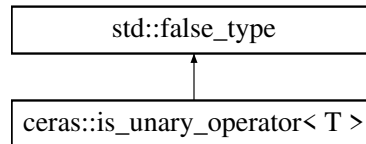
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 >:



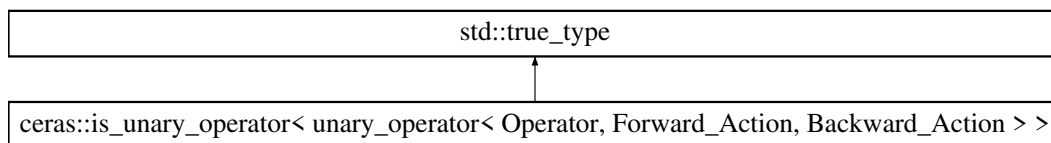
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 > >:



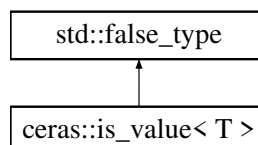
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 >:



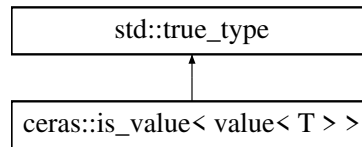
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 > >:



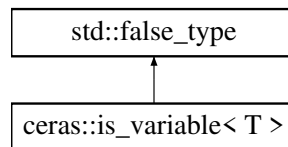
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 >:



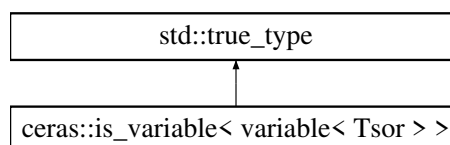
The documentation for this struct was generated from the following file:

- /data/structured_folders/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:

- /data/structured_folders/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 Tsr>
 - auto [predict](#) (Tsr 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
- constexpr [model](#) (Input_List const &input_layer, Output_List const &output_layer)
- constexpr auto [input](#) () const
 - Returns the input layer(s) of the model in a 'list', which is are [place_holders](#).*
- constexpr auto [output](#) () const
 - Returns the output layer(s) of the model in a 'list', which is are expressions.*
- template<Tensor Tsr>
 - constexpr auto [predict](#) (Tsr const &input_tensor) const
- template<List Tsr_List>
 - auto [predict](#) (Tsr_List const &input_tensor) const

Public Attributes

- [output_layer_type](#) [expression_](#)
 - output layer of the model.*
- [input_layer_type](#) [place_holder_](#)
- Input_List [input_layer_](#)
- Output_List [output_layer_](#)

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() [1/2]

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

Parameters

<i>place_holder</i>	The input layer of the model, a place holder.
<i>expression</i>	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 };
```

6.25.3.2 model() [2/2]

```
template<Expression Ex, Place_Holder Ph>
constexpr ceras::model< Ex, Ph >::model (
    Input_List const & input_layer,
    Output_List const & output_layer ) [inline], [constexpr]
```

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() [1/2]

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

Returns the input layer(s) of the model in a 'list', which is are [place_holders](#).

6.25.4.3 input() [2/2]

```
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](#).

6.25.4.4 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

6.25.4.5 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

ex	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
```

6.25.4.6 output() [1/2]

```
template<Expression Ex, Place_Holder Ph>
constexpr auto ceras::model< Ex, Ph >::output ( ) const [inline], [constexpr]
```

Returns the output layer(s) of the model in a 'list', which is are expressions.

6.25.4.7 output() [2/2]

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

Returns the output layer of the model.

6.25.4.8 predict() [1/3]

```
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)
```

6.25.4.9 predict() [2/3]

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

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)
```

6.25.4.10 predict() [3/3]

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

6.25.4.11 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

6.25.4.12 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

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

6.25.4.13 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 input_layer_

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

6.25.5.3 output_layer_

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

6.25.5.4 place_holder_

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

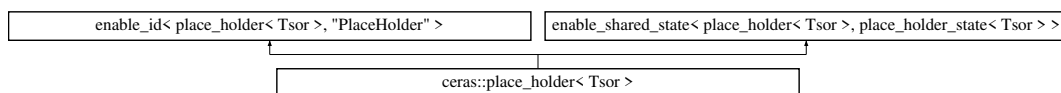
The documentation for this struct was generated from the following files:

- /data/structured_folders/workspace/github.repo/ceras/include/[model.hpp](#)
- /data/structured_folders/workspace/github.repo/ceras/include/[xmodel.hpp](#)

6.26 ceras::place_holder< Tsor > Struct Template Reference

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

Inheritance diagram for ceras::place_holder< Tsor >:



Public Types

- typedef Tsor [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](#) (Tsor data)
- Tsor const [forward](#) () const
- void [reset](#) ()
- void [backward](#) (auto) const noexcept

6.26.1 Member Typedef Documentation

6.26.1.1 tensor_type

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

6.26.2 Constructor & Destructor Documentation

6.26.2.1 place_holder() [1/4]

```
template<Tensor Tsor>
ceras::place\_holder< Tsor >::place\_holder (
    place\_holder< Tsor > 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:

- [/data/structured_folders/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:

- [/data/structured_folders/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_
```

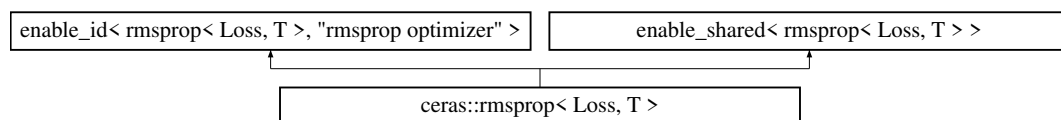
The documentation for this struct was generated from the following file:

- /data/structured_folders/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:

- /data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp

6.30 ceras::ceras_private::session< Tsor > Struct Template Reference

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

Public Types

- typedef [place_holder< Tsor > place_holder_type](#)
- typedef [variable< Tsor > variable_type](#)
- typedef [variable_state< Tsor > 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, Tsor const &value)
- void [bind](#) ([place_holder_type](#) &p_holder, Tsor 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 Tsor>
typedef place\_holder<Tsor> ceras::ceras\_private::session< Tsor >::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_
```

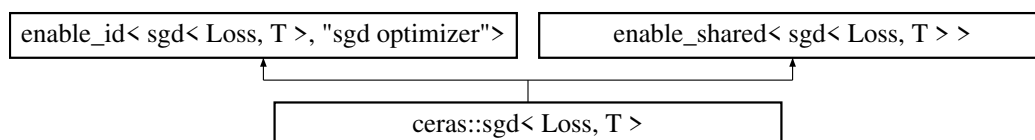
The documentation for this struct was generated from the following file:

- /data/structured_folders/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 nesterov=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_
```

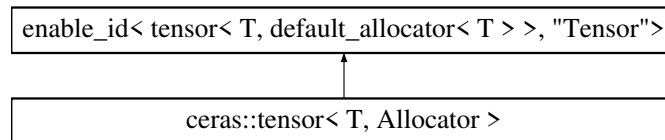
The documentation for this struct was generated from the following file:

- [/data/structured_folders/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:

- /data/structured_folders/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
```

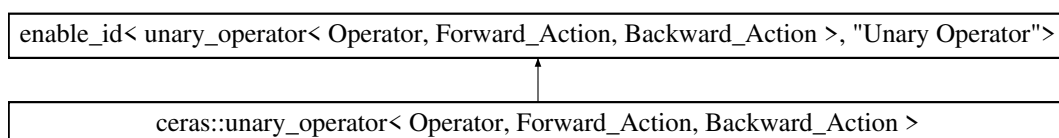
The documentation for this struct was generated from the following file:

- /data/structured_folders/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
```

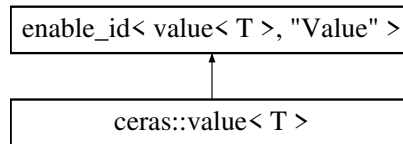
The documentation for this struct was generated from the following file:

- /data/structured_folders/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_
```

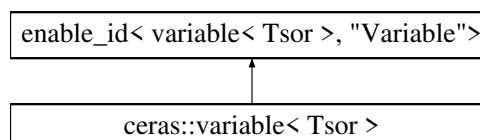
The documentation for this struct was generated from the following file:

- /data/structured_folders/workspace/github.repo/ceras/include/value.hpp

6.36 ceras::variable< Tsor > Struct Template Reference

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

Inheritance diagram for ceras::variable< Tsor >:



Public Types

- typedef Tsor [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 Ttor>
typedef Ttor ceras::variable< Ttor >::tensor_type
```

6.36.1.2 value_type

```
template<Tensor Ttor>
typedef tensor_type::value_type ceras::variable< Ttor >::value_type
```

6.36.2 Constructor & Destructor Documentation

6.36.2.1 variable() [1/4]

```
template<Tensor Tsor>
ceras::variable< Tsor >::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 Tsor>
ceras::variable< Tsor >::variable ( ) [delete]
```

6.36.2.3 variable() [3/4]

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

6.36.2.4 variable() [4/4]

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

6.36.3 Member Function Documentation**6.36.3.1 backward()**

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

6.36.3.2 contexts() [1/2]

```
template<Tensor Tsor>
std::vector<tensor_type>& ceras::variable< Tsor >::contexts ( ) [inline]
```


6.36.3.3 contexts() [2/2]

```
template<Tensor Tzor>
std::vector<tensor_type> ceras::variable< Tzor >::contexts ( ) const [inline]
```

6.36.3.4 data() [1/2]

```
template<Tensor Tzor>
tensor_type& ceras::variable< Tzor >::data ( ) [inline]
```

6.36.3.5 data() [2/2]

```
template<Tensor Tzor>
tensor_type ceras::variable< Tzor >::data ( ) const [inline]
```

6.36.3.6 forward()

```
template<Tensor Tzor>
tensor_type const ceras::variable< Tzor >::forward ( ) [inline], [noexcept]
```

6.36.3.7 gradient() [1/2]

```
template<Tensor Tzor>
tensor_type& ceras::variable< Tzor >::gradient ( ) [inline]
```

6.36.3.8 gradient() [2/2]

```
template<Tensor Tzor>
tensor_type ceras::variable< Tzor >::gradient ( ) const [inline]
```

6.36.3.9 operator=() [1/2]

```
template<Tensor Tzor>
variable& ceras::variable< Tzor >::operator= (
    variable< Tzor > && ) [default]
```

6.36.3.10 operator=() [2/2]

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

6.36.3.11 reset()

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

6.36.3.12 shape()

```
template<Tensor Tsor>
std::vector<std::size_t> ceras::variable< Tsor >::shape ( ) const [inline], [noexcept]
```

6.36.3.13 trainable() [1/2]

```
template<Tensor Tsor>
bool ceras::variable< Tsor >::trainable ( ) const [inline], [noexcept]
```

6.36.3.14 trainable() [2/2]

```
template<Tensor Tsor>
void ceras::variable< Tsor >::trainable (
    bool t ) [inline]
```

6.36.4 Member Data Documentation

6.36.4.1 regularizer_

```
template<Tensor Tsor>
regularizer<value_type> ceras::variable< Tsor >::regularizer_
```

6.36.4.2 state_

```
template<Tensor Tzor>
std::shared_ptr<variable_state<tensor_type> > ceras::variable< Tzor >::state_
```

6.36.4.3 trainable_

```
template<Tensor Tzor>
bool ceras::variable< Tzor >::trainable_
```

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

- /data/structured_folders/workspace/github.repo/ceras/include/[variable.hpp](#)

6.37 ceras::variable_state< Tzor > Struct Template Reference

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

Public Attributes

- Tzor [data_](#)
- Tzor [gradient_](#)
- std::vector< Tzor > [contexts_](#)

6.37.1 Member Data Documentation

6.37.1.1 contexts_

```
template<Tensor Tzor>
std::vector<Tzor> ceras::variable_state< Tzor >::contexts_
```

6.37.1.2 data_

```
template<Tensor Tzor>
Tzor ceras::variable_state< Tzor >::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:

- [/data/structured_folders/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:

- /data/structured_folders/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:

- /data/structured_folders/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:

- /data/structured_folders/workspace/github.repo/ceras/include/[tensor.hpp](#)

Chapter 7

File Documentation

7.1 `/data/structured_`↵ `folders/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. 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>.
- `template<Expression Ex>`
`auto ceras::silu (Ex const &ex) noexcept`
An alias name of activation swish.

7.2 /data/structured_↵ folders/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 /data/structured_↵ folders/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 [/data/structured_↵](#) folders/workspace/github.repo/ceras/include/config.hpp File Reference

7.5 [/data/structured_↵](#) folders/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 >](#)
- 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 /data/structured_folders/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 /data/structured_folders/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 /data/structured_folders/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 /data/structured_↵ folders/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 /data/structured_folders/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 `/data/structured_↔` **folders/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 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 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`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto ceras::minus (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto ceras::operator- (Lhs_Expression const &lhs_ex, Rh_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 Rh_Expression>`
`constexpr auto concatenate (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `auto concatenate (unsigned long axe=-1)`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto concat (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `auto concat (unsigned long axe=-1)`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto maximum (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto atan2 (Lhs_Expression const &lhs_ex, Rh_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 Rhx_Expression>`
`constexpr auto equal (Lhs_Expression const &lhs_ex, Rhx_Expression const &rhs_ex) 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>
constexpr auto equal (
 Lhs_Expression const & lhs_ex,
 Rhs_Expression const & rhs_ex) [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 /data/structured\_folders/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](#) = `adadelat< Loss, T >`

## Variables

- `auto` [ceras::Adam](#)
- `auto` [ceras::SGD](#)
- `auto` [ceras::Adagrad](#)
- `auto` [ceras::RMSprop](#)
- `auto` [ceras::Adadelta](#)

## 7.13 `/data/structured_↵` **folders/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 /data/structured\_folders/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 `/data/structured_↔` **folders/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< 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 Tsor, typename CharT , typename Traits >  
std::basic\_ostream< CharT, Traits > & [ceras::operator<<](#) (std::basic\_ostream< CharT, Traits > &os\_, Tsor 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 Tsor>`  
Tsor `ceras::add` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator+` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator+` (typename Tsor::value\_type const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator+` (Tsor const &lhs, typename Tsor::value\_type const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::minus` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator-` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator-` (typename Tsor::value\_type const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator-` (Tsor const &lhs, typename Tsor::value\_type const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator*` (typename Tsor::value\_type const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator*` (Tsor const &lhs, typename Tsor::value\_type const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator/` (Tsor const &lhs, typename Tsor::value\_type const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::reshape` (Tsor const &ts, std::vector< unsigned long > const &new\_shape)
- `template<Tensor Tsor>`  
void `ceras::multiply` (Tsor const &lhs, Tsor const &rhs, Tsor &ans) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::multiply` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::operator*` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::elementwise_product` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::hadamard_product` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::elementwise_divide` (Tsor const &lhs, Tsor const &rhs) noexcept
- `template<Tensor Tsor>`  
Tsor `ceras::repeat` (Tsor const &tsor, unsigned long n)
- `template<Tensor Tsor>`  
Tsor `ceras::reduce_sum` (Tsor const &tsor)
- `template<Tensor Tsor>`  
Tsor `ceras::reduce_mean` (Tsor const &tsor)
- `template<Tensor Tsor>`  
Tsor `ceras::clip` (Tsor &tsor, typename Tsor::value\_type lower=0, typename Tsor::value\_type upper=1)
- `template<Tensor Tsor>`  
Tsor `ceras::squeeze` (Tsor 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 T>`  
`Tsor ceras::random\_like (Tsor const &tsor, typename T::value_type min=0, typename T::value_type max=1)`
- `template<Tensor T>`  
`Tsor ceras::randn\_like (Tsor const &tsor, typename T::value_type mean=0, typename T::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 T>`  
`Tsor ceras::deep\_copy (Tsor const &tsor)`
- `template<Tensor T>`  
`Tsor ceras::copy (Tsor const &tsor)`
- `template<Tensor T>`  
`Tsor ceras::concatenate (Tsor const &lhs, Tzor const &rhs, unsigned long axis=0) noexcept`
- `template<Tensor T>`  
`Tsor ceras::repmat (Tzor const &tsor, unsigned long row_rep, unsigned long col_rep)`
- `template<Tensor T>`  
`constexpr bool ceras::empty (Tzor const &tsor) noexcept`
- `template<typename T, typename A = default_allocator<T>>>`  
`constexpr tensor< T, A > ceras::zeros (std::vector< unsigned long > const &shape)`
- `template<Tensor T>`  
`constexpr Tzor ceras::zeros\_like (Tzor const &tsor)`
- `template<typename T, typename A = default_allocator<T>>>`  
`constexpr tensor< T, A > ceras::ones (std::vector< unsigned long > const &shape)`
- `template<Tensor T>`  
`constexpr Tzor ceras::ones\_like (Tzor const &tsor)`
- `template<Tensor T>`  
`auto ceras::max (Tzor const &tsor)`
- `template<Tensor T>`  
`auto ceras::amax (Tzor const &tsor)`
- `template<Tensor T>`  
`auto ceras::min (Tzor const &tsor)`
- `template<Tensor T>`  
`auto ceras::amin (Tzor const &tsor)`
- `template<Tensor T>`  
`auto ceras::sum (Tzor const &tsor)`
- `template<Tensor T>`  
`auto ceras::mean (Tzor const &tsor)`
- `template<Tensor T>`  
`auto ceras::norm (Tzor const &tsor)`
- `template<Tensor T>`  
`Tzor ceras::abs (Tzor const &tsor)`
- `template<Tensor T>`  
`Tzor ceras::softmax (Tzor const &tsor)`
- `template<Tensor T>`  
`bool ceras::has\_nan (Tzor const &tsor)`
- `template<Tensor T>`  
`bool ceras::has\_inf (Tzor const &tsor)`
- `template<Tensor T>`  
`bool ceras::is\_valid (Tzor const &tsor)`
- `template<Tensor T, typename Function >`  
`Tzor ceras::reduce (Tzor const &ts, unsigned long axis, typename T::value_type const &init, Function const &func, bool keepdims=false) noexcept`
- `template<Tensor T>`  
`Tzor ceras::sum (Tzor const &ts, unsigned long axis, bool keepdims=false) noexcept`

- `template<Tensor Tsor>`  
requires `std::floating_point< typename Tsor::value_type >` `Tsor` `ceras::mean` (`Tsor` const &ts, unsigned long axis, bool keepdims=false) noexcept
- `template<Tensor Tsor>`  
requires `std::floating_point< typename Tsor::value_type >` `Tsor` `ceras::variance` (`Tsor` const &ts, unsigned long axis, bool keepdims=false) noexcept
- `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
- `template<Tensor Tsor>`  
requires `std::floating_point< typename Tsor::value_type >` `Tsor::value_type` `ceras::var` (`Tsor` const &ts) noexcept
- `template<Tensor Tsor>`  
requires `std::floating_point< typename Tsor::value_type >` `Tsor::value_type` `ceras::std` (`Tsor` const &ts) noexcept
- `template<Tensor Tsor>`  
`Tsor` `ceras::max` (`Tsor` const &ts, unsigned long axis, bool keepdims=false) noexcept
- `template<Tensor Tsor>`  
`Tsor` `ceras::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 >` `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 Tsor>`  
void `ceras::save_tensor` (`std::string` const &file\_name, `Tsor` const &tsor)

## 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 /data/structured\_folders/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 /data/structured\_↔ folders/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< Tzor >](#)
- struct [ceras::regularizer< Float >](#)
- struct [ceras::variable< Tzor >](#)
- struct [ceras::is\\_variable< T >](#)
- struct [ceras::is\\_variable< variable< Tzor > >](#)

## Namespaces

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

## Functions

- template<Tensor Tzor>  
[ceras\\_private::session< Tzor > & 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>

## 7.18 /data/structured\_folders/workspace/github.repo/ceras/include/xmodel.hpp File Reference

```
#include "../includes.hpp"
#include "../operation.hpp"
#include "../place_holder.hpp"
#include "../session.hpp"
#include "../tensor.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/context_cast.hpp"
#include "../utils/tqdm.hpp"
#include "../utils/list.hpp"
#include "../utils/debug.hpp"
```

## Classes

- struct [ceras::model](#)< Ex, Ph >

## Namespaces

- [ceras](#)





# Index

/data/structured\_folders/workspace/github.repo/ceras/include/activation.hpp,  
141 ceras, 18

/data/structured\_folders/workspace/github.repo/ceras/include/adadelta.hpp,  
142 ceras, 51

/data/structured\_folders/workspace/github.repo/ceras/include/adadelta\_complex\_operator.hpp,  
142 ceras::adadelta< Loss, T >, 64

/data/structured\_folders/workspace/github.repo/ceras/include/adagrad.hpp,  
144 ceras, 52

/data/structured\_folders/workspace/github.repo/ceras/include/adagrad\_constant.hpp,  
144 ceras::adagrad< Loss, T >, 66

/data/structured\_folders/workspace/github.repo/ceras/include/adam.hpp,  
145 ceras, 52

/data/structured\_folders/workspace/github.repo/ceras/include/adam\_impl.hpp,  
145 ceras::adam< Loss, T >, 68

/data/structured\_folders/workspace/github.repo/ceras/include/allocator.hpp,  
147 ceras, 20

/data/structured\_folders/workspace/github.repo/ceras/include/loss.hpp,  
148 ceras, 20

/data/structured\_folders/workspace/github.repo/ceras/include/local.hpp,  
149 ceras::tensor< T, Allocator >, 108

/data/structured\_folders/workspace/github.repo/ceras/include/operation.hpp,  
150 ceras, 20

/data/structured\_folders/workspace/github.repo/ceras/include/optimizer.hpp,  
171 ceras, 20

/data/structured\_folders/workspace/github.repo/ceras/include/parameter\_holder.hpp,  
172 ceras::adam< Loss, T >, 68

/data/structured\_folders/workspace/github.repo/ceras/include/session.hpp,  
173 ceras, 20

/data/structured\_folders/workspace/github.repo/ceras/include/tensor.hpp,  
174 ceras::tensor< T, Allocator >, 110

/data/structured\_folders/workspace/github.repo/ceras/include/tensor\_impl.hpp,  
177 ceras, 21

/data/structured\_folders/workspace/github.repo/ceras/include/type.hpp,  
178 ceras::tensor< T, Allocator >, 110

/data/structured\_folders/workspace/github.repo/ceras/include/similarity\_model.hpp,  
179 operation.hpp, 155

~session  
    ceras::ceras\_private::session< T, Allocator >, 102

abs  
    ceras, 19  
    operation.hpp, 154

abs\_loss  
    ceras, 19

acos  
    operation.hpp, 155

acosh  
    operation.hpp, 155

ada\_delta  
    ceras, 18

asinh  
    operation.hpp, 155

atan  
    operation.hpp, 156

atan2  
    operation.hpp, 156

atanh  
    operation.hpp, 156

average\_pooling\_2d  
    operation.hpp, 156

AveragePooling2D  
    ceras, 21

backward

- ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 71
  - ceras::constant< Tsr >, 78
  - ceras::place\_holder< Tsr >, 94
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 118
  - ceras::value< T >, 121
  - ceras::variable< Tsr >, 124
- backward\_action\_
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 71
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 119
- batch\_normalization
  - operation.hpp, 157
- batch\_size\_
  - ceras::view\_4d< T >, 139
- BatchNormalization
  - ceras, 21
- begin
  - ceras::tensor< T, Allocator >, 110
  - ceras::view\_2d< T >, 130
- beta\_1\_
  - ceras::adam< Loss, T >, 68
- beta\_2\_
  - ceras::adam< Loss, T >, 69
- binary\_cross\_entropy\_loss
  - ceras, 22
- Binary\_Operator
  - ceras, 52
- binary\_operator
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 70
- BinaryCrossEntropy
  - ceras, 53
- BinaryCrossentropy
  - ceras, 52
- bind
  - ceras::ceras\_private::session< Tsr >, 102
  - ceras::place\_holder< Tsr >, 95
- CategoricalCrossEntropy
  - ceras, 53
- CategoricalCrossentropy
  - ceras, 53
- cbegin
  - ceras::tensor< T, Allocator >, 111
- cbt
  - operation.hpp, 157
- ceil
  - operation.hpp, 157
- cend
  - ceras::tensor< T, Allocator >, 111
- ceras, 9
  - abs, 19
  - abs\_loss, 19
  - ada\_delta, 18
  - ada\_grad, 18
  - Adadelta, 51
  - Adagrad, 52
  - Adam, 52
  - Add, 20
  - add, 20
  - amax, 20
  - amin, 20
  - arg, 20
  - as\_tensor, 21
  - AveragePooling2D, 21
  - BatchNormalization, 21
  - binary\_cross\_entropy\_loss, 22
  - Binary\_Operator, 52
  - BinaryCrossEntropy, 53
  - BinaryCrossentropy, 52
  - CategoricalCrossEntropy, 53
  - CategoricalCrossentropy, 53
  - clip, 22
  - Complex, 53
  - computation\_graph, 22
  - Concatenate, 23
  - concatenate, 23
  - conj, 23
  - Constant, 54
  - Conv2D, 23
  - copy, 24
  - cross\_entropy, 24
  - cross\_entropy\_loss, 25
  - cube, 18
  - deep\_copy, 25
  - default\_allocator, 18
  - Dense, 25
  - Dropout, 26
  - elementwise\_divide, 26
  - elementwise\_multiply, 26
  - elementwise\_product, 26
  - ELU, 27
  - elu, 26
  - empty, 27
  - exponential, 27
  - Expression, 54
  - Flatten, 27
  - gelu, 27
  - gemm, 27, 28
  - gemm\_cpu, 28
  - get\_default\_session, 28
  - glorot\_uniform, 28
  - hadamard\_product, 28, 29
  - hard\_sigmoid, 29
  - has\_inf, 29
  - has\_nan, 29
  - Hinge, 54
  - hinge\_loss, 29
  - imag, 29
  - Input, 30
  - is\_binary\_operator\_v, 54
  - is\_complex\_v, 54
  - is\_constant\_v, 55
  - is\_place\_holder\_v, 55

- is\_tensor\_v, 55
- is\_unary\_operator\_v, 55
- is\_valid, 30
- is\_value\_v, 55
- is\_variable\_v, 55
- leaky\_relu, 30
- LeakyReLU, 30
- linspace, 30
- load\_tensor, 30
- MAE, 55
- mae, 31
- make\_binary\_operator, 56
- make\_compiled\_model, 31
- make\_trainable, 31
- make\_unary\_operator, 56
- matrix, 18
- max, 31
- MaxPooling2D, 32
- mean, 32
- mean\_absolute\_error, 32
- mean\_reduce, 32
- mean\_squared\_error, 33
- mean\_squared\_logarithmic\_error, 33
- MeanAbsoluteError, 56
- MeanSquaredError, 56
- min, 33
- minus, 33, 34
- MSE, 57
- mse, 34
- Multiply, 34
- multiply, 34
- negative, 35
- negative\_relu, 35
- norm, 35
- ones, 35
- ones\_like, 36
- Operator, 57
- operator!=, 36
- operator<, 41
- operator<<, 41
- operator<=, 41
- operator>, 42
- operator>=, 42
- operator\*, 36, 37
- operator+, 37–39
- operator-, 39, 40
- operator/, 41
- operator==, 41
- Place\_Holder, 57
- plus, 42
- polar, 42
- randn, 43
- randn\_like, 43
- random, 43
- random\_generator, 57
- random\_like, 43
- random\_seed, 58
- read\_tensor, 43
- real, 43
- reduce, 44
- reduce\_mean, 44
- reduce\_sum, 44
- ReLU, 45
- relu, 45
- relu6, 45
- repeat, 45
- replace\_placeholder\_with\_expression, 45
- repmat, 46
- Reshape, 46
- reshape, 46
- rms\_prop, 19
- RMSprop, 58
- save\_tensor, 46
- selu, 46
- SGD, 58
- sigmoid, 47
- silu, 47
- Softmax, 47
- softmax, 47
- softplus, 47
- softsign, 48
- square, 48
- squared\_loss, 48
- squeeze, 48
- standard\_deviation, 49
- std, 49
- Subtract, 49
- sum, 49
- sum\_reduce, 50
- swish, 50
- Tensor, 58
- tesseract, 19
- truncated\_normal, 50
- Unary\_Operator, 58
- update\_cuda\_gemm\_threshold, 50
- UpSampling2D, 50
- Value, 59
- var, 50
- Variable, 59
- variance, 51
- write\_tensor, 51
- zeros, 51
- zeros\_like, 51
- 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, Rhs\_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, 59
- ceras::ceras\_private::session< Tsor >, 100
  - ~session, 102
  - bind, 102
  - deserialize, 102
  - operator=, 102
  - place\_holder\_type, 101
  - place\_holders\_, 104
  - rebind, 102
  - remember, 103
  - restore, 103
  - run, 103
  - save, 103
  - serialize, 103
  - session, 101
  - tap, 103
  - variable\_state\_type, 101
  - variable\_type, 101
  - variables\_, 104
- 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< Tsor >, 78
  - backward, 78
  - constant, 78
  - data\_, 79
  - forward, 79
  - shape, 79
- ceras::dataset, 59
- ceras::dataset::fashion\_mnist, 59
  - load\_data, 59
- ceras::dataset::mnist, 60
  - load\_data, 60
- ceras::gradient\_descent< Loss, T >, 79
  - forward, 80
  - gradient\_descent, 80
  - learning\_rate\_, 80
  - loss\_, 81
  - momentum\_, 81
  - tensor\_type, 80
- ceras::is\_binary\_operator< binary\_operator< Lhs\_Operator, Rhs\_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< Tsor > >, 83
- ceras::is\_constant< T >, 83
- ceras::is\_place\_holder< place\_holder< Tsor > >, 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< Tsor > >, 86
- ceras::model< Ex, Ph >, 87
  - compile, 89
  - expression\_, 92
  - input, 89
  - input\_layer\_, 92
  - input\_layer\_type, 88
  - load\_weights, 89
  - model, 88
  - operator(), 90
  - output, 90
  - output\_layer\_, 93
  - output\_layer\_type, 88
  - place\_holder\_, 93
  - predict, 90, 91

- save\_weights, 92
- summary, 92
- trainable, 92
- ceras::place\_holder< T, S, R >, 93
  - backward, 94
  - bind, 95
  - forward, 95
  - operator=, 95
  - place\_holder, 94
  - reset, 95
  - tensor\_type, 94
- ceras::place\_holder\_state< T, S, R >, 96
  - data\_, 96
  - shape\_hint\_, 96
- ceras::regularizer< Float, T >, 96
  - l1\_, 97
  - l2\_, 97
  - regularizer, 97
  - synchronized\_, 97
  - value\_type, 97
- ceras::rmsprop< Loss, T >, 98
  - decay\_, 99
  - forward, 99
  - iterations\_, 99
  - learning\_rate\_, 99
  - loss\_, 99
  - rho\_, 100
  - rmsprop, 99
  - tensor\_type, 98
- ceras::sgd< Loss, T >, 104
  - decay\_, 106
  - forward, 105
  - iterations\_, 106
  - learning\_rate\_, 106
  - loss\_, 106
  - momentum\_, 106
  - nesterov\_, 106
  - sgd, 105
  - tensor\_type, 105
- ceras::tensor< T, Allocator >, 107
  - allocator, 108
  - as\_scalar, 110
  - as\_type, 110
  - begin, 110
  - cbegin, 111
  - cend, 111
  - copy, 111
  - creep\_to, 111
  - data, 111
  - deep\_copy, 111, 112
  - empty, 112
  - end, 112
  - map, 112
  - memory\_offset\_, 116
  - ndim, 112
  - operator\*=, 113
  - operator+=, 113
  - operator-, 113
  - operator=, 113, 114
  - operator/=: 114
  - operator=, 114
  - operator[], 114, 115
  - reset, 115
  - reshape, 115
  - resize, 115
  - self\_type, 108
  - shape, 115
  - shape\_, 116
  - shared\_vector, 108
  - shrink\_to, 116
  - size, 116
  - slice, 116
  - tensor, 109, 110
  - value\_type, 108
  - vector\_, 116
  - vector\_type, 109
- ceras::tensor\_deduction< L, R >, 117
  - op\_type, 117
  - tensor\_type, 117
- ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 117
  - backward, 118
  - backward\_action\_, 119
  - forward, 118
  - forward\_action\_, 119
  - input\_data\_, 119
  - op\_, 119
  - output\_data\_, 119
  - tensor\_type, 119
  - unary\_operator, 118
- ceras::value< T >, 120
  - backward, 121
  - data\_, 122
  - forward, 121
  - operator=, 121, 122
  - value, 120, 121
  - value\_type, 120
- ceras::variable< T, S, R >, 122
  - backward, 124
  - contexts, 124
  - data, 125
  - forward, 125
  - gradient, 125
  - operator=, 125
  - regularizer\_, 126
  - reset, 126
  - shape, 126
  - state\_, 126
  - tensor\_type, 123
  - trainable, 126
  - trainable\_, 127
  - value\_type, 123
  - variable, 123, 124
- ceras::variable\_state< T, S, R >, 127
  - contexts\_, 127
  - data\_, 127

- gradient\_, 127
- ceras::view\_2d< T >, 128
  - begin, 130
  - col, 130
  - col\_, 133
  - col\_begin, 130, 131
  - col\_end, 131
  - col\_type, 129
  - const\_col\_type, 129
  - const\_row\_type, 129
  - data, 131
  - data\_, 133
  - end, 131
  - operator[], 132
  - row, 132
  - row\_, 133
  - row\_begin, 132
  - row\_end, 132, 133
  - row\_type, 129
  - shape, 133
  - size, 133
  - transposed\_, 134
  - value\_type, 129
  - view\_2d, 129, 130
- ceras::view\_3d< T >, 134
  - channel\_, 135
  - col\_, 135
  - data\_, 135
  - operator[], 135
  - row\_, 135
  - view\_3d, 134
- ceras::view\_4d< T >, 136
  - batch\_size\_, 139
  - channel\_, 139
  - col\_, 139
  - data\_, 139
  - operator[], 138
  - row\_, 139
  - view\_4d, 136
- channel\_
  - ceras::view\_3d< T >, 135
  - ceras::view\_4d< T >, 139
- clip
  - ceras, 22
  - operation.hpp, 157
- col
  - ceras::view\_2d< T >, 130
- col\_
  - ceras::view\_2d< T >, 133
  - ceras::view\_3d< T >, 135
  - ceras::view\_4d< T >, 139
- col\_begin
  - ceras::view\_2d< T >, 130, 131
- col\_end
  - ceras::view\_2d< T >, 131
- col\_type
  - ceras::view\_2d< T >, 129
- compile
  - ceras::model< Ex, Ph >, 89
- compiled\_model
  - ceras::compiled\_model< Model, Optimizer, Loss >, 73
- compiled\_optimizer\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, 76
- Complex
  - ceras, 53
- computation\_graph
  - ceras, 22
- concat
  - operation.hpp, 157, 158
- Concatenate
  - ceras, 23
- concatenate
  - ceras, 23
  - operation.hpp, 158
- conj
  - ceras, 23
- const\_col\_type
  - ceras::view\_2d< T >, 129
- const\_row\_type
  - ceras::view\_2d< T >, 129
- Constant
  - ceras, 54
- constant
  - ceras::constant< T, Allocator >, 78
- contexts
  - ceras::variable< T, Allocator >, 124
- contexts\_
  - ceras::variable\_state< T, Allocator >, 127
- Conv2D
  - ceras, 23
- conv2d
  - operation.hpp, 158
- copy
  - ceras, 24
  - ceras::tensor< T, Allocator >, 111
- cos
  - operation.hpp, 158
- cosh
  - operation.hpp, 159
- creep\_to
  - ceras::tensor< T, Allocator >, 111
- cross\_entropy
  - ceras, 24
- cross\_entropy\_loss
  - ceras, 25
- cube
  - ceras, 18
- data
  - ceras::tensor< T, Allocator >, 111
  - ceras::variable< T, Allocator >, 125
  - ceras::view\_2d< T >, 131
- data\_
  - ceras::constant< T, Allocator >, 79
  - ceras::place\_holder\_state< T, Allocator >, 96

- ceras::value< T >, [122](#)
  - ceras::variable\_state< Tsor >, [127](#)
  - ceras::view\_2d< T >, [133](#)
  - ceras::view\_3d< T >, [135](#)
  - ceras::view\_4d< T >, [139](#)
- decay\_
  - ceras::adagrad< Loss, T >, [66](#)
  - ceras::rmsprop< Loss, T >, [99](#)
  - ceras::sgd< Loss, T >, [106](#)
- deep\_copy
  - ceras, [25](#)
  - ceras::tensor< T, Allocator >, [111](#), [112](#)
- default\_allocator
  - ceras, [18](#)
- Dense
  - ceras, [25](#)
- deserialize
  - ceras::ceras\_private::session< Tsor >, [102](#)
- drop\_out
  - operation.hpp, [159](#)
- Dropout
  - ceras, [26](#)
- elementwise\_divide
  - ceras, [26](#)
- elementwise\_multiply
  - ceras, [26](#)
- elementwise\_product
  - ceras, [26](#)
- ELU
  - ceras, [27](#)
- elu
  - ceras, [26](#)
- empty
  - ceras, [27](#)
  - ceras::tensor< T, Allocator >, [112](#)
- end
  - ceras::tensor< T, Allocator >, [112](#)
  - ceras::view\_2d< T >, [131](#)
- equal
  - operation.hpp, [159](#)
- erf
  - operation.hpp, [160](#)
- erfc
  - operation.hpp, [160](#)
- evaluate
  - ceras::compiled\_model< Model, Optimizer, Loss >, [74](#)
- exp
  - operation.hpp, [160](#)
- exp2
  - operation.hpp, [160](#)
- expm1
  - operation.hpp, [161](#)
- exponential
  - ceras, [27](#)
- Expression
  - ceras, [54](#)
- expression\_
  - ceras::model< Ex, Ph >, [92](#)
- fabs
  - operation.hpp, [161](#)
- fit
  - ceras::compiled\_model< Model, Optimizer, Loss >, [74](#)
- Flatten
  - ceras, [27](#)
- flatten
  - operation.hpp, [161](#)
- floor
  - operation.hpp, [161](#)
- forward
  - ceras::adadelta< Loss, T >, [64](#)
  - ceras::adagrad< Loss, T >, [66](#)
  - ceras::adam< Loss, T >, [68](#)
  - ceras::binary\_operator< Lhs\_Operator, Rhc\_Operator, Forward\_Action, Backward\_Action >, [71](#)
  - ceras::constant< Tsor >, [79](#)
  - ceras::gradient\_descent< Loss, T >, [80](#)
  - ceras::place\_holder< Tsor >, [95](#)
  - ceras::rmsprop< Loss, T >, [99](#)
  - ceras::sgd< Loss, T >, [105](#)
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, [118](#)
  - ceras::value< T >, [121](#)
  - ceras::variable< Tsor >, [125](#)
- forward\_action\_
  - ceras::binary\_operator< Lhs\_Operator, Rhc\_Operator, Forward\_Action, Backward\_Action >, [71](#)
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, [119](#)
- gelu
  - ceras, [27](#)
- gemm
  - ceras, [27](#), [28](#)
- gemm\_cpu
  - ceras, [28](#)
- get\_default\_session
  - ceras, [28](#)
- glorot\_uniform
  - ceras, [28](#)
- gradient
  - ceras::variable< Tsor >, [125](#)
- gradient\_
  - ceras::variable\_state< Tsor >, [127](#)
- gradient\_descent
  - ceras::gradient\_descent< Loss, T >, [80](#)
- ground\_truth\_place\_holder\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, [76](#)
- hadamard\_product
  - ceras, [28](#), [29](#)
- hard\_sigmoid
  - ceras, [29](#)
- has\_inf

- ceras, 29
- has\_nan
  - ceras, 29
- Hinge
  - ceras, 54
- hinge\_loss
  - ceras, 29
- hypot
  - operation.hpp, 162
- identity
  - operation.hpp, 162
- imag
  - ceras, 29
- imag\_
  - ceras::complex< Real\_Ex, Imag\_Ex >, 77
- img2col
  - operation.hpp, 162
- includes.hpp
  - STB\_IMAGE\_IMPLEMENTATION, 146
  - STB\_IMAGE\_RESIZE\_IMPLEMENTATION, 146
  - STB\_IMAGE\_WRITE\_IMPLEMENTATION, 146
- Input
  - ceras, 30
- input
  - ceras::model< Ex, Ph >, 89
- input\_data\_
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 119
- input\_layer\_
  - ceras::model< Ex, Ph >, 92
- input\_layer\_type
  - ceras::model< Ex, Ph >, 88
- 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, 54
- is\_complex\_v
  - ceras, 54
- is\_constant\_v
  - ceras, 55
- is\_place\_holder\_v
  - ceras, 55
- is\_tensor\_v
  - ceras, 55
- is\_unary\_operator\_v
  - ceras, 55
- is\_valid
  - ceras, 30
- is\_value\_v
  - ceras, 55
- is\_variable\_v
  - ceras, 55
- iterations\_
  - ceras::adadelta< Loss, T >, 64
  - ceras::adagrad< Loss, T >, 66
  - ceras::adam< Loss, T >, 69
  - ceras::rmsprop< Loss, T >, 99
  - ceras::sgd< Loss, T >, 106
- l1\_
  - ceras::regularizer< Float >, 97
- l2\_
  - ceras::regularizer< Float >, 97
- leaky\_relu
  - ceras, 30
- LeakyReLU
  - ceras, 30
- learning\_rate\_
  - ceras::adadelta< Loss, T >, 64
  - ceras::adagrad< Loss, T >, 66
  - ceras::adam< Loss, T >, 69
  - ceras::gradient\_descent< Loss, T >, 80
  - ceras::rmsprop< Loss, T >, 99
  - ceras::sgd< Loss, T >, 106
- lhs\_input\_data\_
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 71
- lhs\_op\_
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 71
- linspace
  - ceras, 30
- llrint
  - operation.hpp, 162
- llround
  - operation.hpp, 162
- load\_data
  - ceras::dataset::fashion\_mnist, 59
  - ceras::dataset::mnist, 60
- load\_tensor
  - ceras, 30
- load\_weights
  - ceras::model< Ex, Ph >, 89
- log
  - operation.hpp, 163
- log10
  - operation.hpp, 163
- log1p
  - operation.hpp, 163
- log2
  - operation.hpp, 163
- 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 >, 99
  - ceras::sgd< Loss, T >, 106
- lrint
  - operation.hpp, 164
- lround



- operation.hpp, 164
- MAE
  - ceras, 55
- mae
  - ceras, 31
- make\_binary\_operator
  - ceras, 56
- make\_compiled\_model
  - ceras, 31
- make\_trainable
  - ceras, 31
- make\_unary\_operator
  - ceras, 56
- map
  - ceras::tensor< T, Allocator >, 112
- matrix
  - ceras, 18
- max
  - ceras, 31
- max\_pooling\_2d
  - operation.hpp, 164
- maximum
  - operation.hpp, 164
- MaxPooling2D
  - ceras, 32
- mean
  - ceras, 32
- mean\_absolute\_error
  - ceras, 32
- mean\_reduce
  - ceras, 32
- mean\_squared\_error
  - ceras, 33
- mean\_squared\_logarithmic\_error
  - ceras, 33
- MeanAbsoluteError
  - ceras, 56
- MeanSquaredError
  - ceras, 56
- memory\_offset\_
  - ceras::tensor< T, Allocator >, 116
- min
  - ceras, 33
- minus
  - ceras, 33, 34
- 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 >, 106
- MSE
  - ceras, 57
- mse
  - ceras, 34
- Multiply
  - ceras, 34
- multiply
  - ceras, 34
- ndim
  - ceras::tensor< T, Allocator >, 112
- nearbyint
  - operation.hpp, 164
- negative
  - ceras, 35
- negative\_relu
  - ceras, 35
- nesterov\_
  - ceras::sgd< Loss, T >, 106
- norm
  - ceras, 35
- normalization\_batch
  - operation.hpp, 165
- ones
  - ceras, 35
- ones\_like
  - ceras, 36
- operation.hpp, 165
- op\_
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 119
- op\_type
  - ceras::tensor\_deduction< L, R >, 117
- operation.hpp
  - abs, 154
  - acos, 155
  - acosh, 155
  - asin, 155
  - asinh, 155
  - atan, 156
  - atan2, 156
  - atanh, 156
  - average\_pooling\_2d, 156
  - batch\_normalization, 157
  - cbrt, 157
  - ceil, 157
  - clip, 157
  - concat, 157, 158
  - concatenate, 158
  - conv2d, 158
  - cos, 158
  - cosh, 159
  - drop\_out, 159
  - equal, 159
  - erf, 160
  - erfc, 160
  - exp, 160
  - exp2, 160
  - expm1, 161
  - fabs, 161
  - flatten, 161
  - floor, 161
  - hypot, 162

- identity, 162
- img2col, 162
- llrint, 162
- llround, 162
- log, 163
- log10, 163
- log1p, 163
- log2, 163
- lrint, 164
- lround, 164
- max\_pooling\_2d, 164
- maximum, 164
- nearbyint, 164
- normalization\_batch, 165
- ones\_like, 165
- random\_normal\_like, 165
- reduce\_max, 166
- reduce\_min, 166
- reduce\_sum, 166
- repeat, 167
- reshape, 167
- rint, 167
- round, 167
- sign, 168
- sin, 168
- sinh, 168
- sqr, 171
- sqrt, 169
- tan, 169
- tanh, 169
- transpose, 169
- trunc, 170
- up\_sampling\_2d, 170
- y, 171
- zero\_padding\_2d, 170
- zeros\_like, 170
- Operator
  - ceras, 57
- operator!=
  - ceras, 36
- operator<
  - ceras, 41
- operator<<
  - ceras, 41
- operator<=
  - ceras, 41
- operator>
  - ceras, 42
- operator>=
  - ceras, 42
- operator\*
  - ceras, 36, 37
- operator\*=
  - ceras::tensor< T, Allocator >, 113
- operator()
  - ceras::compiled\_model< Model, Optimizer, Loss >, 75
  - ceras::model< Ex, Ph >, 90
- operator+
  - ceras, 37–39
- operator+=
  - ceras::tensor< T, Allocator >, 113
- operator-
  - ceras, 39, 40
  - ceras::tensor< T, Allocator >, 113
- operator-=
  - ceras::tensor< T, Allocator >, 113, 114
- operator/
  - ceras, 41
- operator/=
  - ceras::tensor< T, Allocator >, 114
- operator=
  - ceras::ceras\_private::session< Tzor >, 102
  - ceras::place\_holder< Tzor >, 95
  - ceras::tensor< T, Allocator >, 114
  - ceras::value< T >, 121, 122
  - ceras::variable< Tzor >, 125
- operator==
  - ceras, 41
- operator[]
  - ceras::tensor< T, Allocator >, 114, 115
  - ceras::view\_2d< T >, 132
  - ceras::view\_3d< T >, 135
  - ceras::view\_4d< T >, 138
- 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, Rhz\_Operator, Forward\_Action, Backward\_Action >, 72
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 119
- output\_layer\_
  - ceras::model< Ex, Ph >, 93
- output\_layer\_type
  - ceras::model< Ex, Ph >, 88
- Place\_Holder
  - ceras, 57
- place\_holder
  - ceras::place\_holder< Tzor >, 94
- place\_holder\_
  - ceras::model< Ex, Ph >, 93
- place\_holder\_type
  - ceras::ceras\_private::session< Tzor >, 101
- place\_holders\_
  - ceras::ceras\_private::session< Tzor >, 104
- plus
  - ceras, 42
- polar
  - ceras, 42
- predict

- ceras::compiled\_model< Model, Optimizer, Loss >, [75](#)
  - ceras::model< Ex, Ph >, [90](#), [91](#)
- randn
  - ceras, [43](#)
- randn\_like
  - ceras, [43](#)
- random
  - ceras, [43](#)
- random\_generator
  - ceras, [57](#)
- random\_like
  - ceras, [43](#)
- random\_normal\_like
  - operation.hpp, [165](#)
- random\_seed
  - ceras, [58](#)
- read\_tensor
  - ceras, [43](#)
- real
  - ceras, [43](#)
- real\_
  - ceras::complex< Real\_Ex, Imag\_Ex >, [77](#)
- rebind
  - ceras::ceras\_private::session< Tzor >, [102](#)
- reduce
  - ceras, [44](#)
- reduce\_max
  - operation.hpp, [166](#)
- reduce\_mean
  - ceras, [44](#)
- reduce\_min
  - operation.hpp, [166](#)
- reduce\_sum
  - ceras, [44](#)
  - operation.hpp, [166](#)
- regularizer
  - ceras::regularizer< Float >, [97](#)
- regularizer\_
  - ceras::variable< Tzor >, [126](#)
- ReLU
  - ceras, [45](#)
- relu
  - ceras, [45](#)
- relu6
  - ceras, [45](#)
- remember
  - ceras::ceras\_private::session< Tzor >, [103](#)
- repeat
  - ceras, [45](#)
  - operation.hpp, [167](#)
- replace\_placeholder\_with\_expression
  - ceras, [45](#)
- repmat
  - ceras, [46](#)
- reset
  - ceras::place\_holder< Tzor >, [95](#)
  - ceras::tensor< T, Allocator >, [115](#)
- ceras::variable< Tzor >, [126](#)
- Reshape
  - ceras, [46](#)
- reshape
  - ceras, [46](#)
  - ceras::tensor< T, Allocator >, [115](#)
  - operation.hpp, [167](#)
- resize
  - ceras::tensor< T, Allocator >, [115](#)
- restore
  - ceras::ceras\_private::session< Tzor >, [103](#)
- rho\_
  - ceras::adadelta< Loss, T >, [65](#)
  - ceras::rmsprop< Loss, T >, [100](#)
- 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, [167](#)
- rms\_prop
  - ceras, [19](#)
- RMSprop
  - ceras, [58](#)
- rmsprop
  - ceras::rmsprop< Loss, T >, [99](#)
- round
  - operation.hpp, [167](#)
- row
  - ceras::view\_2d< T >, [132](#)
- row\_
  - ceras::view\_2d< T >, [133](#)
  - ceras::view\_3d< T >, [135](#)
  - ceras::view\_4d< T >, [139](#)
- row\_begin
  - ceras::view\_2d< T >, [132](#)
- row\_end
  - ceras::view\_2d< T >, [132](#), [133](#)
- row\_type
  - ceras::view\_2d< T >, [129](#)
- run
  - ceras::ceras\_private::session< Tzor >, [103](#)
- save
  - ceras::ceras\_private::session< Tzor >, [103](#)
- save\_tensor
  - ceras, [46](#)
- save\_weights
  - ceras::model< Ex, Ph >, [92](#)
- self\_type
  - ceras::tensor< T, Allocator >, [108](#)
- selu
  - ceras, [46](#)
- serialize
  - ceras::ceras\_private::session< Tzor >, [103](#)
- session
  - ceras::ceras\_private::session< Tzor >, [101](#)

- SGD
  - ceras, [58](#)
- sgd
  - ceras::sgd< Loss, T >, [105](#)
- shape
  - ceras::constant< T, Allocator >, [79](#)
  - ceras::tensor< T, Allocator >, [115](#)
  - ceras::variable< T, Allocator >, [126](#)
  - ceras::view\_2d< T >, [133](#)
- shape\_
  - ceras::tensor< T, Allocator >, [116](#)
- shape\_hint\_
  - ceras::place\_holder\_state< T, Allocator >, [96](#)
- shared\_vector
  - ceras::tensor< T, Allocator >, [108](#)
- shrink\_to
  - ceras::tensor< T, Allocator >, [116](#)
- sigmoid
  - ceras, [47](#)
- sign
  - operation.hpp, [168](#)
- silu
  - ceras, [47](#)
- sin
  - operation.hpp, [168](#)
- sinh
  - operation.hpp, [168](#)
- size
  - ceras::tensor< T, Allocator >, [116](#)
  - ceras::view\_2d< T >, [133](#)
- slice
  - ceras::tensor< T, Allocator >, [116](#)
- Softmax
  - ceras, [47](#)
- softmax
  - ceras, [47](#)
- softplus
  - ceras, [47](#)
- softsign
  - ceras, [48](#)
- sqr
  - operation.hpp, [171](#)
- sqrt
  - operation.hpp, [169](#)
- square
  - ceras, [48](#)
- squared\_loss
  - ceras, [48](#)
- squeeze
  - ceras, [48](#)
- standard\_deviation
  - ceras, [49](#)
- state\_
  - ceras::variable< T, Allocator >, [126](#)
- STB\_IMAGE\_IMPLEMENTATION
  - includes.hpp, [146](#)
- STB\_IMAGE\_RESIZE\_IMPLEMENTATION
  - includes.hpp, [146](#)
- STB\_IMAGE\_WRITE\_IMPLEMENTATION
  - includes.hpp, [146](#)
- std
  - ceras, [49](#)
- Subtract
  - ceras, [49](#)
- sum
  - ceras, [49](#)
- sum\_reduce
  - ceras, [50](#)
- summary
  - ceras::model< Ex, Ph >, [92](#)
- swish
  - ceras, [50](#)
- synchronized\_
  - ceras::regularizer< Float >, [97](#)
- tan
  - operation.hpp, [169](#)
- tanh
  - operation.hpp, [169](#)
- tap
  - ceras::ceras\_private::session< T, Allocator >, [103](#)
- Tensor
  - ceras, [58](#)
- tensor
  - ceras::tensor< T, Allocator >, [109](#), [110](#)
- 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 >, [94](#)
  - ceras::rmsprop< Loss, T >, [98](#)
  - ceras::sgd< Loss, T >, [105](#)
  - ceras::tensor\_deduction< L, R >, [117](#)
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, [119](#)
  - ceras::variable< T, Allocator >, [123](#)
- 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 >, [92](#)
  - ceras::variable< T, Allocator >, [126](#)
- trainable\_
  - ceras::variable< T, Allocator >, [127](#)
- transpose
  - operation.hpp, [169](#)
- transposed\_
  - ceras::view\_2d< T >, [134](#)
- trunc
  - operation.hpp, [170](#)

- truncated\_normal
  - ceras, 50
- Unary\_Operator
  - ceras, 58
- unary\_operator
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 118
- up\_sampling\_2d
  - operation.hpp, 170
- update\_cuda\_gemm\_threshold
  - ceras, 50
- UpSampling2D
  - ceras, 50
- Value
  - ceras, 59
- value
  - ceras::value< T >, 120, 121
- value\_type
  - ceras::regularizer< Float >, 97
  - ceras::tensor< T, Allocator >, 108
  - ceras::value< T >, 120
  - ceras::variable< Tsor >, 123
  - ceras::view\_2d< T >, 129
- var
  - ceras, 50
- Variable
  - ceras, 59
- variable
  - ceras::variable< Tsor >, 123, 124
- variable\_state\_type
  - ceras::ceras\_private::session< Tsor >, 101
- variable\_type
  - ceras::ceras\_private::session< Tsor >, 101
- variables\_
  - ceras::ceras\_private::session< Tsor >, 104
- variance
  - ceras, 51
- vector\_
  - ceras::tensor< T, Allocator >, 116
- vector\_type
  - ceras::tensor< T, Allocator >, 109
- view\_2d
  - ceras::view\_2d< T >, 129, 130
- view\_3d
  - ceras::view\_3d< T >, 134
- view\_4d
  - ceras::view\_4d< T >, 136
- write\_tensor
  - ceras, 51
- y
  - operation.hpp, 171
- zero\_padding\_2d
  - operation.hpp, 170
- zeros
  - ceras, 51
  - zeros\_like
    - ceras, 51
  - operation.hpp, 170