

ceras

Generated by Doxygen 1.9.1



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 ceras Namespace Reference	9
5.1.1 Typedef Documentation	17
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	18
5.1.1.6 rms_prop	18
5.1.1.7 tesseract	18
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()	19
5.1.2.4 Add()	19
5.1.2.5 add()	19
5.1.2.6 amax()	20
5.1.2.7 amin()	20
5.1.2.8 as_tensor()	20
5.1.2.9 average_pooling_2d()	20
5.1.2.10 AveragePooling2D()	20
5.1.2.11 batch_normalization()	20
5.1.2.12 BatchNormalization() [1/2]	21
5.1.2.13 BatchNormalization() [2/2]	21
5.1.2.14 binary_cross_entropy_loss()	21
5.1.2.15 clip() [1/2]	22
5.1.2.16 clip() [2/2]	22
5.1.2.17 computation_graph()	22
5.1.2.18 concat() [1/2]	22
5.1.2.19 concat() [2/2]	22
5.1.2.20 concatenate() [1/3]	23
5.1.2.21 concatenate() [2/3]	23

5.1.2.22 concatenate() [3/3]	23
5.1.2.23 Concatenate()	23
5.1.2.24 Conv2D()	23
5.1.2.25 conv2d()	24
5.1.2.26 copy()	24
5.1.2.27 cross_entropy()	25
5.1.2.28 cross_entropy_loss()	25
5.1.2.29 deep_copy()	25
5.1.2.30 Dense()	25
5.1.2.31 drop_out()	26
5.1.2.32 Dropout()	26
5.1.2.33 elementwise_divide()	26
5.1.2.34 elementwise_multiply()	26
5.1.2.35 elementwise_product() [1/2]	26
5.1.2.36 elementwise_product() [2/2]	27
5.1.2.37 elu()	27
5.1.2.38 ELU()	27
5.1.2.39 empty()	27
5.1.2.40 equal()	27
5.1.2.41 exp()	28
5.1.2.42 exponential()	28
5.1.2.43 Flatten()	28
5.1.2.44 flatten()	28
5.1.2.45 gelu()	28
5.1.2.46 gemm() [1/2]	29
5.1.2.47 gemm() [2/2]	29
5.1.2.48 gemm_cpu()	29
5.1.2.49 get_default_session()	29
5.1.2.50 glorot_uniform()	29
5.1.2.51 hadamard_product() [1/2]	30
5.1.2.52 hadamard_product() [2/2]	30
5.1.2.53 hard_sigmoid()	30
5.1.2.54 has_inf()	30
5.1.2.55 has_nan()	30
5.1.2.56 hinge_loss()	30
5.1.2.57 identity()	31
5.1.2.58 img2col()	31
5.1.2.59 Input()	31
5.1.2.60 is_valid()	31
5.1.2.61 leaky_relu()	31
5.1.2.62 LeakyReLU()	31
5.1.2.63 linspace()	32

5.1.2.64 <code>load_tensor()</code>	32
5.1.2.65 <code>log()</code>	32
5.1.2.66 <code>mae()</code>	32
5.1.2.67 <code>make_compiled_model()</code>	32
5.1.2.68 <code>make_trainable()</code>	33
5.1.2.69 <code>max()</code> [1/2]	33
5.1.2.70 <code>max()</code> [2/2]	33
5.1.2.71 <code>max_pooling_2d()</code>	33
5.1.2.72 <code>maximum()</code>	33
5.1.2.73 <code>MaxPooling2D()</code>	33
5.1.2.74 <code>mean()</code> [1/2]	34
5.1.2.75 <code>mean()</code> [2/2]	34
5.1.2.76 <code>mean_absolute_error()</code>	34
5.1.2.77 <code>mean_reduce()</code>	34
5.1.2.78 <code>mean_squared_error()</code>	34
5.1.2.79 <code>mean_squared_logarithmic_error()</code>	35
5.1.2.80 <code>min()</code> [1/2]	35
5.1.2.81 <code>min()</code> [2/2]	35
5.1.2.82 <code>minus()</code> [1/2]	35
5.1.2.83 <code>minus()</code> [2/2]	35
5.1.2.84 <code>mse()</code>	36
5.1.2.85 <code>Multiply()</code>	36
5.1.2.86 <code>multiply()</code> [1/2]	36
5.1.2.87 <code>multiply()</code> [2/2]	36
5.1.2.88 <code>negative()</code>	36
5.1.2.89 <code>negative_relu()</code>	37
5.1.2.90 <code>norm()</code>	37
5.1.2.91 <code>normalization_batch()</code>	37
5.1.2.92 <code>ones()</code>	37
5.1.2.93 <code>ones_like()</code> [1/2]	37
5.1.2.94 <code>ones_like()</code> [2/2]	38
5.1.2.95 <code>operator"!=()</code>	38
5.1.2.96 <code>operator*()</code> [1/4]	38
5.1.2.97 <code>operator*()</code> [2/4]	38
5.1.2.98 <code>operator*()</code> [3/4]	38
5.1.2.99 <code>operator*()</code> [4/4]	38
5.1.2.100 <code>operator+()</code> [1/4]	39
5.1.2.101 <code>operator+()</code> [2/4]	39
5.1.2.102 <code>operator+()</code> [3/4]	39
5.1.2.103 <code>operator+()</code> [4/4]	39
5.1.2.104 <code>operator-()</code> [1/4]	39
5.1.2.105 <code>operator-()</code> [2/4]	39

5.1.2.106 operator-() [3/4]	40
5.1.2.107 operator-() [4/4]	40
5.1.2.108 operator/()	40
5.1.2.109 operator<()	40
5.1.2.110 operator<<()	40
5.1.2.111 operator<=()	40
5.1.2.112 operator==(1/2)	41
5.1.2.113 operator==(2/2)	41
5.1.2.114 operator>()	41
5.1.2.115 operator>=()	41
5.1.2.116 plus()	41
5.1.2.117 randn()	42
5.1.2.118 randn_like()	42
5.1.2.119 random()	42
5.1.2.120 random_like()	42
5.1.2.121 random_normal_like()	42
5.1.2.122 read_tensor()	43
5.1.2.123 reduce()	43
5.1.2.124 reduce_mean(1/2)	43
5.1.2.125 reduce_mean(2/2)	43
5.1.2.126 reduce_sum(1/2)	44
5.1.2.127 reduce_sum(2/2)	44
5.1.2.128 relu()	44
5.1.2.129 ReLU()	44
5.1.2.130 repeat()	44
5.1.2.131 replace_placeholder_with_expression()	44
5.1.2.132 repmat()	45
5.1.2.133 Reshape()	45
5.1.2.134 reshape(1/2)	45
5.1.2.135 reshape(2/2)	45
5.1.2.136 save_tensor()	46
5.1.2.137 selu()	46
5.1.2.138 sigmoid()	46
5.1.2.139 sign()	46
5.1.2.140 Softmax()	46
5.1.2.141 softmax(1/2)	47
5.1.2.142 softmax(2/2)	47
5.1.2.143 softplus()	47
5.1.2.144 softsign()	47
5.1.2.145 square()	47
5.1.2.146 squared_loss()	48
5.1.2.147 squeeze()	48

5.1.2.148 standard_deviation()	48
5.1.2.149 std()	48
5.1.2.150 Subtract()	48
5.1.2.151 sum() <sup>[1/2]</sup>	49
5.1.2.152 sum() <sup>[2/2]</sup>	49
5.1.2.153 sum_reduce()	49
5.1.2.154 tanh()	49
5.1.2.155 transpose()	49
5.1.2.156 truncated_normal()	49
5.1.2.157 up_sampling_2d()	50
5.1.2.158 update_cuda_gemm_threshold()	50
5.1.2.159 UpSampling2D()	50
5.1.2.160 var()	50
5.1.2.161 variance()	50
5.1.2.162 write_tensor()	50
5.1.2.163 zero_padding_2d()	50
5.1.2.164 zeros()	51
5.1.2.165 zeros_like()	51
5.1.3 Variable Documentation	51
5.1.3.1 Adadelta	51
5.1.3.2 Adagrad	52
5.1.3.3 Adam	52
5.1.3.4 Binary_Operator	52
5.1.3.5 BinaryCrossentropy	52
5.1.3.6 BinaryCrossEntropy	53
5.1.3.7 CategoricalCrossentropy	53
5.1.3.8 CategoricalCrossEntropy	53
5.1.3.9 Constant	53
5.1.3.10 Expression	53
5.1.3.11 Hinge	54
5.1.3.12 is_binary_operator_v	54
5.1.3.13 is_constant_v	54
5.1.3.14 is_place_holder_v	54
5.1.3.15 is_tensor_v	54
5.1.3.16 is_unary_operator_v	54
5.1.3.17 is_value_v	55
5.1.3.18 is_variable_v	55
5.1.3.19 MAE	55
5.1.3.20 make_binary_operator	55
5.1.3.21 make_unary_operator	55
5.1.3.22 MeanAbsoluteError	56
5.1.3.23 MeanSquaredError	56

5.1.3.24 MSE	56
5.1.3.25 Operator	57
5.1.3.26 Place_Holder	57
5.1.3.27 random_generator	57
5.1.3.28 random_seed	57
5.1.3.29 RMSprop	57
5.1.3.30 SGD	57
5.1.3.31 Tensor	58
5.1.3.32 Unary_Operator	58
5.1.3.33 Value	58
5.1.3.34 Variable	58
5.2 ceras::ceras_private Namespace Reference	58
5.3 ceras::dataset Namespace Reference	58
5.4 ceras::dataset::fashion_mnist Namespace Reference	58
5.4.1 Function Documentation	59
5.4.1.1 load_data()	59
5.5 ceras::dataset::mnist Namespace Reference	59
5.5.1 Function Documentation	59
5.5.1.1 load_data()	59
<b>6 Class Documentation</b>	<b>61</b>
6.1 ceras::adadelat< Loss, T > Struct Template Reference	61
6.1.1 Member Typedef Documentation	61
6.1.1.1 tensor_type	62
6.1.2 Constructor & Destructor Documentation	62
6.1.2.1 adadelat()	62
6.1.3 Member Function Documentation	62
6.1.3.1 forward()	62
6.1.4 Member Data Documentation	62
6.1.4.1 iterations_	62
6.1.4.2 learning_rate_	62
6.1.4.3 loss_	63
6.1.4.4 rho_	63
6.2 ceras::adagrad< Loss, T > Struct Template Reference	63
6.2.1 Member Typedef Documentation	63
6.2.1.1 tensor_type	64
6.2.2 Constructor & Destructor Documentation	64
6.2.2.1 adagrad()	64
6.2.3 Member Function Documentation	64
6.2.3.1 forward()	64
6.2.4 Member Data Documentation	64
6.2.4.1 decay_	64



6.2.4.2 iterations_ . . . . .	64
6.2.4.3 learning_rate_ . . . . .	65
6.2.4.4 loss_ . . . . .	65
6.3 ceras::adam< Loss, T > Struct Template Reference . . . . .	65
6.3.1 Member Typedef Documentation . . . . .	66
6.3.1.1 tensor_type . . . . .	66
6.3.2 Constructor & Destructor Documentation . . . . .	66
6.3.2.1 adam() . . . . .	66
6.3.3 Member Function Documentation . . . . .	66
6.3.3.1 forward() . . . . .	66
6.3.4 Member Data Documentation . . . . .	66
6.3.4.1 amsgrad_ . . . . .	66
6.3.4.2 beta_1_ . . . . .	67
6.3.4.3 beta_2_ . . . . .	67
6.3.4.4 iterations_ . . . . .	67
6.3.4.5 learning_rate_ . . . . .	67
6.3.4.6 loss_ . . . . .	67
6.4 ceras::binary_operator< Lhs_Operator, Rh_Operator, Forward_Action, Backward_Action > Struct Template Reference . . . . .	67
6.4.1 Member Typedef Documentation . . . . .	68
6.4.1.1 tensor_type . . . . .	68
6.4.2 Constructor & Destructor Documentation . . . . .	68
6.4.2.1 binary_operator() . . . . .	68
6.4.3 Member Function Documentation . . . . .	69
6.4.3.1 backward() . . . . .	69
6.4.3.2 forward() . . . . .	69
6.4.4 Member Data Documentation . . . . .	69
6.4.4.1 backward_action_ . . . . .	69
6.4.4.2 forward_action_ . . . . .	69
6.4.4.3 lhs_input_data_ . . . . .	69
6.4.4.4 lhs_op_ . . . . .	70
6.4.4.5 output_data_ . . . . .	70
6.4.4.6 rhs_input_data_ . . . . .	70
6.4.4.7 rhs_op_ . . . . .	70
6.5 ceras::compiled_model< Model, Optimizer, Loss > Struct Template Reference . . . . .	70
6.5.1 Member Typedef Documentation . . . . .	71
6.5.1.1 io_layer_type . . . . .	71
6.5.2 Constructor & Destructor Documentation . . . . .	71
6.5.2.1 compiled_model() . . . . .	71
6.5.3 Member Function Documentation . . . . .	72
6.5.3.1 evaluate() . . . . .	72
6.5.3.2 fit() . . . . .	72

6.5.3.3 operator()	73
6.5.3.4 predict()	73
6.5.3.5 train_on_batch()	73
6.5.3.6 trainable()	74
6.5.4 Member Data Documentation	74
6.5.4.1 compiled_optimizer_	74
6.5.4.2 ground_truth_place_holder_	74
6.5.4.3 input_place_holder_	74
6.5.4.4 loss_	74
6.5.4.5 model_	75
6.5.4.6 optimizer_	75
6.5.4.7 optimizer_type	75
6.6 ceras::constant< Tsor > Struct Template Reference	75
6.6.1 Constructor & Destructor Documentation	76
6.6.1.1 constant()	76
6.6.2 Member Function Documentation	76
6.6.2.1 backward()	76
6.6.2.2 forward()	76
6.6.2.3 shape()	76
6.6.3 Member Data Documentation	76
6.6.3.1 data_	76
6.7 ceras::gradient_descent< Loss, T > Struct Template Reference	77
6.7.1 Member Typedef Documentation	77
6.7.1.1 tensor_type	77
6.7.2 Constructor & Destructor Documentation	77
6.7.2.1 gradient_descent()	77
6.7.3 Member Function Documentation	78
6.7.3.1 forward()	78
6.7.4 Member Data Documentation	78
6.7.4.1 learning_rate_	78
6.7.4.2 loss_	78
6.7.4.3 momentum_	78
6.8 ceras::is_binary_operator< T > Struct Template Reference	78
6.9 ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > > Struct Template Reference	79
6.10 ceras::is_constant< T > Struct Template Reference	79
6.11 ceras::is_constant< constant< Tsor > > Struct Template Reference	79
6.12 ceras::is_place_holder< T > Struct Template Reference	80
6.13 ceras::is_place_holder< place_holder< Tsor > > Struct Template Reference	80
6.14 ceras::is_tensor< T > Struct Template Reference	80
6.15 ceras::is_tensor< tensor< T, A > > Struct Template Reference	81
6.16 ceras::is_unary_operator< T > Struct Template Reference	81

6.17 ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > > Struct Template Reference	81
6.18 ceras::is_value< T > Struct Template Reference	82
6.19 ceras::is_value< value< T > > Struct Template Reference	82
6.20 ceras::is_variable< T > Struct Template Reference	82
6.21 ceras::is_variable< variable< Tsor > > Struct Template Reference	83
6.22 ceras::model< Ex, Ph > Struct Template Reference	83
6.22.1 Detailed Description	83
6.22.2 Member Typedef Documentation	84
6.22.2.1 input_layer_type	84
6.22.2.2 output_layer_type	84
6.22.3 Constructor & Destructor Documentation	84
6.22.3.1 model()	84
6.22.4 Member Function Documentation	84
6.22.4.1 compile()	85
6.22.4.2 input()	85
6.22.4.3 load_weights()	85
6.22.4.4 operator>()	85
6.22.4.5 output()	86
6.22.4.6 predict()	86
6.22.4.7 save_weights()	87
6.22.4.8 summary()	87
6.22.4.9 trainable()	87
6.22.5 Member Data Documentation	87
6.22.5.1 expression_	87
6.22.5.2 place_holder_	87
6.23 ceras::place_holder< Tsor > Struct Template Reference	88
6.23.1 Member Typedef Documentation	88
6.23.1.1 tensor_type	88
6.23.2 Constructor & Destructor Documentation	88
6.23.2.1 place_holder() [1/4]	88
6.23.2.2 place_holder() [2/4]	89
6.23.2.3 place_holder() [3/4]	89
6.23.2.4 place_holder() [4/4]	89
6.23.3 Member Function Documentation	89
6.23.3.1 backward()	89
6.23.3.2 bind()	89
6.23.3.3 forward()	89
6.23.3.4 operator=() [1/2]	90
6.23.3.5 operator=() [2/2]	90
6.23.3.6 reset()	90
6.24 ceras::place_holder_state< Tsor > Struct Template Reference	90

6.24.1 Member Data Documentation	90
6.24.1.1 data_	90
6.24.1.2 shape_hint_	91
6.25 ceras::regularizer< Float > Struct Template Reference	91
6.25.1 Member Typedef Documentation	91
6.25.1.1 value_type	91
6.25.2 Constructor & Destructor Documentation	91
6.25.2.1 regularizer()	92
6.25.3 Member Data Documentation	92
6.25.3.1 l1_	92
6.25.3.2 l2_	92
6.25.3.3 synchronized_	92
6.26 ceras::rmsprop< Loss, T > Struct Template Reference	92
6.26.1 Member Typedef Documentation	93
6.26.1.1 tensor_type	93
6.26.2 Constructor & Destructor Documentation	93
6.26.2.1 rmsprop()	93
6.26.3 Member Function Documentation	93
6.26.3.1 forward()	94
6.26.4 Member Data Documentation	94
6.26.4.1 decay_	94
6.26.4.2 iterations_	94
6.26.4.3 learning_rate_	94
6.26.4.4 loss_	94
6.26.4.5 rho_	94
6.27 ceras::ceras_private::session< Tsor > Struct Template Reference	95
6.27.1 Member Typedef Documentation	95
6.27.1.1 place_holder_type	95
6.27.1.2 variable_state_type	96
6.27.1.3 variable_type	96
6.27.2 Constructor & Destructor Documentation	96
6.27.2.1 session() [1/3]	96
6.27.2.2 session() [2/3]	96
6.27.2.3 session() [3/3]	96
6.27.2.4 ~session()	96
6.27.3 Member Function Documentation	97
6.27.3.1 bind()	97
6.27.3.2 deserialize()	97
6.27.3.3 operator=() [1/2]	97
6.27.3.4 operator=() [2/2]	97
6.27.3.5 rebind()	97
6.27.3.6 remember()	98

6.27.3.7 restore()	98
6.27.3.8 run()	98
6.27.3.9 save()	98
6.27.3.10 serialize()	98
6.27.3.11 tap()	98
6.27.4 Member Data Documentation	99
6.27.4.1 place_holders_	99
6.27.4.2 variables_	99
6.28 ceras::sgd< Loss, T > Struct Template Reference	99
6.28.1 Member Typedef Documentation	100
6.28.1.1 tensor_type	100
6.28.2 Constructor & Destructor Documentation	100
6.28.2.1 sgd()	100
6.28.3 Member Function Documentation	100
6.28.3.1 forward()	100
6.28.4 Member Data Documentation	100
6.28.4.1 decay_	101
6.28.4.2 iterations_	101
6.28.4.3 learning_rate_	101
6.28.4.4 loss_	101
6.28.4.5 momentum_	101
6.28.4.6 nesterov_	101
6.29 ceras::tensor< T, Allocator > Struct Template Reference	102
6.29.1 Member Typedef Documentation	103
6.29.1.1 allocator	103
6.29.1.2 self_type	103
6.29.1.3 shared_vector	103
6.29.1.4 value_type	104
6.29.1.5 vector_type	104
6.29.2 Constructor & Destructor Documentation	104
6.29.2.1 tensor() [1/7]	104
6.29.2.2 tensor() [2/7]	104
6.29.2.3 tensor() [3/7]	104
6.29.2.4 tensor() [4/7]	104
6.29.2.5 tensor() [5/7]	105
6.29.2.6 tensor() [6/7]	105
6.29.2.7 tensor() [7/7]	105
6.29.3 Member Function Documentation	105
6.29.3.1 as_scalar()	105
6.29.3.2 as_type()	105
6.29.3.3 begin() [1/2]	105
6.29.3.4 begin() [2/2]	106

6.29.3.5 cbegin()	106
6.29.3.6 cend()	106
6.29.3.7 copy()	106
6.29.3.8 creep_to()	106
6.29.3.9 data() [1/2]	106
6.29.3.10 data() [2/2]	106
6.29.3.11 deep_copy() [1/2]	107
6.29.3.12 deep_copy() [2/2]	107
6.29.3.13 empty()	107
6.29.3.14 end() [1/2]	107
6.29.3.15 end() [2/2]	107
6.29.3.16 map()	107
6.29.3.17 ndim()	108
6.29.3.18 operator*=( ) [1/2]	108
6.29.3.19 operator*=( ) [2/2]	108
6.29.3.20 operator+=( ) [1/2]	108
6.29.3.21 operator+=( ) [2/2]	108
6.29.3.22 operator-( )	108
6.29.3.23 operator-=( ) [1/2]	109
6.29.3.24 operator-=( ) [2/2]	109
6.29.3.25 operator/=( ) [1/2]	109
6.29.3.26 operator/=( ) [2/2]	109
6.29.3.27 operator=( ) [1/2]	109
6.29.3.28 operator=( ) [2/2]	109
6.29.3.29 operator[]() [1/2]	110
6.29.3.30 operator[]() [2/2]	110
6.29.3.31 reset()	110
6.29.3.32 reshape()	110
6.29.3.33 resize()	110
6.29.3.34 shape()	111
6.29.3.35 shrink_to()	111
6.29.3.36 size()	111
6.29.3.37 slice()	111
6.29.4 Member Data Documentation	111
6.29.4.1 memory_offset_	111
6.29.4.2 shape_	111
6.29.4.3 vector_	112
6.30 ceras::tensor_deduction< L, R > Struct Template Reference	112
6.30.1 Member Typedef Documentation	112
6.30.1.1 op_type	112
6.30.1.2 tensor_type	112
6.31 ceras::unary_operator< Operator, Forward_Action, Backward_Action > Struct Template Reference	112

6.31.1 Constructor & Destructor Documentation	113
6.31.1.1 unary_operator()	113
6.31.2 Member Function Documentation	113
6.31.2.1 backward()	113
6.31.2.2 forward()	113
6.31.3 Member Data Documentation	114
6.31.3.1 backward_action_	114
6.31.3.2 forward_action_	114
6.31.3.3 input_data_	114
6.31.3.4 op_	114
6.31.3.5 output_data_	114
6.31.3.6 tensor_type	114
6.32 ceras::value< T > Struct Template Reference	115
6.32.1 Member Typedef Documentation	115
6.32.1.1 value_type	115
6.32.2 Constructor & Destructor Documentation	115
6.32.2.1 value() [1/4]	116
6.32.2.2 value() [2/4]	116
6.32.2.3 value() [3/4]	116
6.32.2.4 value() [4/4]	116
6.32.3 Member Function Documentation	116
6.32.3.1 backward()	116
6.32.3.2 forward()	116
6.32.3.3 operator=() [1/2]	117
6.32.3.4 operator=() [2/2]	117
6.32.4 Member Data Documentation	117
6.32.4.1 data_	117
6.33 ceras::variable< Tsor > Struct Template Reference	117
6.33.1 Member Typedef Documentation	118
6.33.1.1 tensor_type	118
6.33.1.2 value_type	118
6.33.2 Constructor & Destructor Documentation	118
6.33.2.1 variable() [1/4]	119
6.33.2.2 variable() [2/4]	119
6.33.2.3 variable() [3/4]	119
6.33.2.4 variable() [4/4]	119
6.33.3 Member Function Documentation	119
6.33.3.1 backward()	119
6.33.3.2 contexts() [1/2]	119
6.33.3.3 contexts() [2/2]	120
6.33.3.4 data() [1/2]	120
6.33.3.5 data() [2/2]	120

6.33.3.6 forward()	120
6.33.3.7 gradient() [1/2]	120
6.33.3.8 gradient() [2/2]	120
6.33.3.9 operator=() [1/2]	120
6.33.3.10 operator=() [2/2]	121
6.33.3.11 reset()	121
6.33.3.12 shape()	121
6.33.3.13 trainable() [1/2]	121
6.33.3.14 trainable() [2/2]	121
6.33.4 Member Data Documentation	121
6.33.4.1 regularizer_	121
6.33.4.2 state_	122
6.33.4.3 trainable_	122
6.34 ceras::variable_state< Tsor > Struct Template Reference	122
6.34.1 Member Data Documentation	122
6.34.1.1 contexts_	122
6.34.1.2 data_	122
6.34.1.3 gradient_	123
6.35 ceras::view_2d< T > Struct Template Reference	123
6.35.1 Constructor & Destructor Documentation	123
6.35.1.1 view_2d() [1/3]	123
6.35.1.2 view_2d() [2/3]	124
6.35.1.3 view_2d() [3/3]	124
6.35.2 Member Function Documentation	124
6.35.2.1 data() [1/2]	124
6.35.2.2 data() [2/2]	124
6.35.2.3 operator[]()	124
6.35.2.4 operator[]() [2/2]	125
6.35.2.5 shape()	125
6.35.2.6 size()	125
6.35.3 Member Data Documentation	125
6.35.3.1 col_	125
6.35.3.2 data_	125
6.35.3.3 row_	125
6.35.3.4 transposed_	126
6.36 ceras::view_3d< T > Struct Template Reference	126
6.36.1 Constructor & Destructor Documentation	126
6.36.1.1 view_3d()	126
6.36.2 Member Function Documentation	126
6.36.2.1 operator[]()	127
6.36.2.2 operator[]() [2/2]	127
6.36.3 Member Data Documentation	127



6.36.3.1 channel_ . . . . .	127
6.36.3.2 col_ . . . . .	127
6.36.3.3 data_ . . . . .	127
6.36.3.4 row_ . . . . .	127
6.37 ceras::view_4d< T > Struct Template Reference . . . . .	128
6.37.1 Detailed Description . . . . .	128
6.37.2 Constructor & Destructor Documentation . . . . .	128
6.37.2.1 view_4d() . . . . .	128
6.37.3 Member Function Documentation . . . . .	129
6.37.3.1 operator[]() [1/2] . . . . .	129
6.37.3.2 operator[]() [2/2] . . . . .	129
6.37.4 Member Data Documentation . . . . .	130
6.37.4.1 batch_size_ . . . . .	130
6.37.4.2 channel_ . . . . .	130
6.37.4.3 col_ . . . . .	130
6.37.4.4 data_ . . . . .	130
6.37.4.5 row_ . . . . .	130
<b>7 File Documentation</b> . . . . .	<b>131</b>
7.1 /data/structured_folders/workspace/github.repo/ceras/include/activation.hpp File Reference . . . . .	131
7.2 /data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp File Reference . . . . .	132
7.3 /data/structured_folders/workspace/github.repo/ceras/include/config.hpp File Reference . . . . .	132
7.4 /data/structured_folders/workspace/github.repo/ceras/include/constant.hpp File Reference . . . . .	132
7.5 /data/structured_folders/workspace/github.repo/ceras/include/dataset.hpp File Reference . . . . .	133
7.6 /data/structured_folders/workspace/github.repo/ceras/include/includes.hpp File Reference . . . . .	134
7.6.1 Macro Definition Documentation . . . . .	134
7.6.1.1 STB_IMAGE_IMPLEMENTATION . . . . .	135
7.6.1.2 STB_IMAGE_RESIZE_IMPLEMENTATION . . . . .	135
7.6.1.3 STB_IMAGE_WRITE_IMPLEMENTATION . . . . .	135
7.7 /data/structured_folders/workspace/github.repo/ceras/include/layer.hpp File Reference . . . . .	135
7.8 /data/structured_folders/workspace/github.repo/ceras/include/loss.hpp File Reference . . . . .	136
7.9 /data/structured_folders/workspace/github.repo/ceras/include/model.hpp File Reference . . . . .	137
7.10 /data/structured_folders/workspace/github.repo/ceras/include/operation.hpp File Reference . . . . .	138
7.11 /data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp File Reference . . . . .	141
7.12 /data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp File Reference . . . . .	142
7.13 /data/structured_folders/workspace/github.repo/ceras/include/session.hpp File Reference . . . . .	143
7.14 /data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp File Reference . . . . .	143
7.15 /data/structured_folders/workspace/github.repo/ceras/include/value.hpp File Reference . . . . .	147
7.16 /data/structured_folders/workspace/github.repo/ceras/include/variable.hpp File Reference . . . . .	148
<b>Index</b> . . . . .	<b>149</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">ceras</a>	9
<a href="#">ceras::ceras_private</a>	58
<a href="#">ceras::dataset</a>	58
<a href="#">ceras::dataset::fashion_mnist</a>	58
<a href="#">ceras::dataset::mnist</a>	59



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

ceras::compiled_model< Model, Optimizer, Loss > . . . . .	70
enable_id	
ceras::adadelta< Loss, T > . . . . .	61
ceras::adagrad< Loss, T > . . . . .	63
ceras::adam< Loss, T > . . . . .	65
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > . . . . .	67
ceras::constant< Tsor > . . . . .	75
ceras::gradient_descent< Loss, T > . . . . .	77
ceras::place_holder< Tsor > . . . . .	88
ceras::rmsprop< Loss, T > . . . . .	92
ceras::sgd< Loss, T > . . . . .	99
ceras::tensor< T, Allocator > . . . . .	102
ceras::unary_operator< Operator, Forward_Action, Backward_Action > . . . . .	112
ceras::value< T > . . . . .	115
ceras::variable< Tsor > . . . . .	117
enable_shared	
ceras::adadelta< Loss, T > . . . . .	61
ceras::adagrad< Loss, T > . . . . .	63
ceras::adam< Loss, T > . . . . .	65
ceras::gradient_descent< Loss, T > . . . . .	77
ceras::rmsprop< Loss, T > . . . . .	92
ceras::sgd< Loss, T > . . . . .	99
enable_shared_state	
ceras::place_holder< Tsor > . . . . .	88
std::false_type	
ceras::is_binary_operator< T > . . . . .	78
ceras::is_constant< T > . . . . .	79
ceras::is_place_holder< T > . . . . .	80
ceras::is_tensor< T > . . . . .	80
ceras::is_unary_operator< T > . . . . .	81
ceras::is_value< T > . . . . .	82
ceras::is_variable< T > . . . . .	82
ceras::model< Ex, Ph > . . . . .	83
ceras::place_holder_state< Tsor > . . . . .	90
ceras::regularizer< Float > . . . . .	91

ceras::regularizer< value_type > . . . . .	91
ceras::ceras_private::session< Tsor > . . . . .	95
ceras::tensor_deduction< L, R > . . . . .	112
ceras::tensor_deduction< Lhs_Operator, Rhs_Operator > . . . . .	112
std::true_type	
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > > . . . . .	79
ceras::is_constant< constant< Tsor > > . . . . .	79
ceras::is_place_holder< place_holder< Tsor > > . . . . .	80
ceras::is_tensor< tensor< T, A > > . . . . .	81
ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > > . . . . .	81
ceras::is_value< value< T > > . . . . .	82
ceras::is_variable< variable< Tsor > > . . . . .	83
ceras::variable_state< Tsor > . . . . .	122
ceras::view_2d< T > . . . . .	123
ceras::view_3d< T > . . . . .	126
ceras::view_4d< T > . . . . .	128

## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">ceras::adadelat&lt; Loss, T &gt;</a>	<a href="#">61</a>
<a href="#">ceras::adagrad&lt; Loss, T &gt;</a>	<a href="#">63</a>
<a href="#">ceras::adam&lt; Loss, T &gt;</a>	<a href="#">65</a>
<a href="#">ceras::binary_operator&lt; Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action &gt;</a>	<a href="#">67</a>
<a href="#">ceras::compiled_model&lt; Model, Optimizer, Loss &gt;</a>	<a href="#">70</a>
<a href="#">ceras::constant&lt; Tsor &gt;</a>	<a href="#">75</a>
<a href="#">ceras::gradient_descent&lt; Loss, T &gt;</a>	<a href="#">77</a>
<a href="#">ceras::is_binary_operator&lt; T &gt;</a>	<a href="#">78</a>
<a href="#">ceras::is_binary_operator&lt; binary_operator&lt; Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action &gt; &gt;</a>	<a href="#">79</a>
<a href="#">ceras::is_constant&lt; T &gt;</a>	<a href="#">79</a>
<a href="#">ceras::is_constant&lt; constant&lt; Tsor &gt; &gt;</a>	<a href="#">79</a>
<a href="#">ceras::is_place_holder&lt; T &gt;</a>	<a href="#">80</a>
<a href="#">ceras::is_place_holder&lt; place_holder&lt; Tsor &gt; &gt;</a>	<a href="#">80</a>
<a href="#">ceras::is_tensor&lt; T &gt;</a>	<a href="#">80</a>
<a href="#">ceras::is_tensor&lt; tensor&lt; T, A &gt; &gt;</a>	<a href="#">81</a>
<a href="#">ceras::is_unary_operator&lt; T &gt;</a>	<a href="#">81</a>
<a href="#">ceras::is_unary_operator&lt; unary_operator&lt; Operator, Forward_Action, Backward_Action &gt; &gt;</a>	<a href="#">81</a>
<a href="#">ceras::is_value&lt; T &gt;</a>	<a href="#">82</a>
<a href="#">ceras::is_value&lt; value&lt; T &gt; &gt;</a>	<a href="#">82</a>
<a href="#">ceras::is_variable&lt; T &gt;</a>	<a href="#">82</a>
<a href="#">ceras::is_variable&lt; variable&lt; Tsor &gt; &gt;</a>	<a href="#">83</a>
<a href="#">ceras::model&lt; Ex, Ph &gt;</a>	<a href="#">83</a>
<a href="#">ceras::place_holder&lt; Tsor &gt;</a>	<a href="#">88</a>
<a href="#">ceras::place_holder_state&lt; Tsor &gt;</a>	<a href="#">90</a>
<a href="#">ceras::regularizer&lt; Float &gt;</a>	<a href="#">91</a>
<a href="#">ceras::rmsprop&lt; Loss, T &gt;</a>	<a href="#">92</a>
<a href="#">ceras::ceras_private::session&lt; Tsor &gt;</a>	<a href="#">95</a>
<a href="#">ceras::sgd&lt; Loss, T &gt;</a>	<a href="#">99</a>
<a href="#">ceras::tensor&lt; T, Allocator &gt;</a>	<a href="#">102</a>
<a href="#">ceras::tensor_deduction&lt; L, R &gt;</a>	<a href="#">112</a>
<a href="#">ceras::unary_operator&lt; Operator, Forward_Action, Backward_Action &gt;</a>	<a href="#">112</a>
<a href="#">ceras::value&lt; T &gt;</a>	<a href="#">115</a>
<a href="#">ceras::variable&lt; Tsor &gt;</a>	<a href="#">117</a>
<a href="#">ceras::variable_state&lt; Tsor &gt;</a>	<a href="#">122</a>
<a href="#">ceras::view_2d&lt; T &gt;</a>	<a href="#">123</a>
<a href="#">ceras::view_3d&lt; T &gt;</a>	<a href="#">126</a>
<a href="#">ceras::view_4d&lt; T &gt;</a>	<a href="#">128</a>





## 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 . . . . .	131
/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp . . . . .	132
/data/structured_folders/workspace/github.repo/ceras/include/config.hpp . . . . .	132
/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp . . . . .	132
/data/structured_folders/workspace/github.repo/ceras/include/dataset.hpp . . . . .	133
/data/structured_folders/workspace/github.repo/ceras/include/includes.hpp . . . . .	134
/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp . . . . .	135
/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp . . . . .	136
/data/structured_folders/workspace/github.repo/ceras/include/model.hpp . . . . .	137
/data/structured_folders/workspace/github.repo/ceras/include/operation.hpp . . . . .	138
/data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp . . . . .	141
/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp . . . . .	142
/data/structured_folders/workspace/github.repo/ceras/include/session.hpp . . . . .	143
/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp . . . . .	143
/data/structured_folders/workspace/github.repo/ceras/include/value.hpp . . . . .	147
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp . . . . .	148



## Chapter 5

# Namespace Documentation

### 5.1 ceras Namespace Reference

#### Namespaces

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

#### Classes

- 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 [adadelat](#)
- struct [adam](#)
- struct [gradient\\_descent](#)
- struct [place\\_holder\\_state](#)
- struct [place\\_holder](#)
- struct [is\\_place\\_holder](#)
- struct [is\\_place\\_holder< place\\_holder< Tsor > >](#)
- struct [tensor](#)
- struct [is\\_tensor](#)
- struct [is\\_tensor< tensor< T, A > >](#)
- struct [view\\_2d](#)
- struct [view\\_3d](#)
- struct [view\\_4d](#)

- struct [value](#)
- struct [is\\_value](#)
- struct [is\\_value< value< T > >](#)
- struct [tensor\\_deduction](#)
- struct [variable\\_state](#)
- struct [regularizer](#)
- struct [variable](#)
- struct [is\\_variable](#)
- struct [is\\_variable< variable< Tsor > >](#)

## Typedefs

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

## Functions

- template<Expression Ex>  
constexpr auto [softmax](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [selu](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [softplus](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [softsign](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [sigmoid](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [tanh](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [relu](#) (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`
- `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 Rhx_Expression>`  
`constexpr auto mean\_squared\_logarithmic\_error (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhx_Expression>`  
`constexpr auto squared\_loss (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhx_Expression>`  
`constexpr auto mean\_squared\_error (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhx_Expression>`  
`constexpr auto mse (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhx_Expression>`  
`constexpr auto abs\_loss (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhx_Expression>`  
`constexpr auto mean\_absolute\_error (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhx_Expression>`  
`constexpr auto mae (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept`

- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto cross\_entropy (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto binary\_cross\_entropy\_loss (Lhs_Expression const &ground_truth, Rhs_Expression const &prediction) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto cross\_entropy\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto hinge\_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`void make\_trainable (Ex &ex, bool t)`
- `template<Expression Ex, Place_Holder Ph, Expression Ey>`  
`auto replace\_placeholder\_with\_expression (Ex const &ex, Ph const &old_place_holder, Ey const &new_↵  
expression)`
- `template<typename Model , typename Optimizer , typename Loss >`  
`auto make\_compiled\_model (Model const &m, Loss const &l, Optimizer const &o)`
- `template<Expression Ex>`  
`std::string computation\_graph (Ex const &ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto plus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto operator+ (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`auto operator\* (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto log (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto negative (Ex const &ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto elementwise\_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto elementwise\_multiply (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto hadamard\_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto sum\_reduce (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto reduce\_sum (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto mean\_reduce (Ex const &ex) noexcept`  
*Computes the mean of elements across all dimensions of an expression.*
- `template<Expression Ex>`  
`constexpr auto reduce\_mean (Ex const &ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto minus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto operator- (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto square (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto abs (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`constexpr auto exp (Ex const &ex) 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`, unsigned long const `col_stride=1`, unsigned long const `row_dilation=1`, unsigned long const `col_dilation=1`) noexcept
- `auto conv2d` (unsigned long `row_input`, unsigned long `col_input`, unsigned long const `row_stride=1`, unsigned long const `col_stride=1`, unsigned long const `row_dilation=1`, unsigned long const `col_dilation=1`, `std::string const &padding="valid"`) noexcept
- `template<typename T >`  
requires `std::floating_point< T > auto drop_out` (T const factor) noexcept
- `auto max_pooling_2d` (unsigned long stride) noexcept
- `auto average_pooling_2d` (unsigned long stride) noexcept
- `auto up_sampling_2d` (unsigned long stride) noexcept
- `template<typename T = double>`  
requires `std::floating_point< T > auto normalization_batch` (T const momentum=0.98) noexcept
- `template<typename T >`  
requires `std::floating_point< T > auto batch_normalization` (T const momentum=0.98) noexcept
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto concatenate` (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- `auto concatenate` (unsigned long `axe=-1`)
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto concat` (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- `auto concat` (unsigned long `axe=-1`)
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto maximum` (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- `template<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 Lhs_Expression, Expression Rhs_Expression>`  
`constexpr auto equal` (Lhs\_Expression const &lhs\_ex, Rhs\_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).*
- `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<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<Expression Ex>
constexpr auto ceras::abs (
    Ex const & ex ) [constexpr], [noexcept]
```

### 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 as\_tensor()

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

#### 5.1.2.9 average\_pooling\_2d()

```
auto ceras::average_pooling_2d (
    unsigned long stride ) [inline], [noexcept]
```

#### 5.1.2.10 AveragePooling2D()

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

Average pooling operation for spatial data.

#### 5.1.2.11 batch\_normalization()

```
template<typename T >
requires std::floating_point<T> auto ceras::batch_normalization (
    T const momentum = 0.98 ) [inline], [noexcept]
```

**5.1.2.12 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]
```

**5.1.2.13 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.
```

**5.1.2.14 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]
```

**5.1.2.15 clip() [1/2]**

```
template<typename Float >
requires std::floating_point<Float> constexpr auto ceras::clip (
    Float lower,
    Float upper = std::numeric_limits<Float>::max() ) [constexpr], [noexcept]
```

**5.1.2.16 clip() [2/2]**

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

**5.1.2.17 computation\_graph()**

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

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

**Parameters**

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

**Returns**

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

**5.1.2.18 concat() [1/2]**

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

**5.1.2.19 concat() [2/2]**

```
auto ceras::concat (
    unsigned long axe = -1 ) [inline]
```



**5.1.2.20 concatenate()** [1/3]

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

**5.1.2.21 concatenate()** [2/3]

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

**5.1.2.22 concatenate()** [3/3]

```
auto ceras::concatenate (
    unsigned long axe = -1 ) [inline]
```

**5.1.2.23 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)
```

**5.1.2.24 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 };

```

#### 5.1.2.25 conv2d()

```

auto ceras::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]

```

#### 5.1.2.26 copy()

```

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

```

**5.1.2.27 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]
```

**5.1.2.28 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]
```

**5.1.2.29 deep\_copy()**

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

**5.1.2.30 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 };
```

#### 5.1.2.31 drop\_out()

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

#### 5.1.2.32 Dropout()

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

Applies Dropout to the input.

#### 5.1.2.33 elementwise\_divide()

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

#### 5.1.2.34 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]
```

#### 5.1.2.35 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]
```

**5.1.2.36 elementwise\_product()** [2/2]

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

**5.1.2.37 elu()**

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

**5.1.2.38 ELU()**

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

Exponential Linear Unit.

**5.1.2.39 empty()**

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

**5.1.2.40 equal()**

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::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);
```

**5.1.2.41 exp()**

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

**5.1.2.42 exponential()**

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

**5.1.2.43 Flatten()**

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

Flattens the input. Does not affect the batch size.

**5.1.2.44 flatten()**

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

**5.1.2.45 gelu()**

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

#### 5.1.2.46 `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 )
```

#### 5.1.2.47 `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 )
```

#### 5.1.2.48 `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 )
```

#### 5.1.2.49 `get_default_session()`

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

#### 5.1.2.50 `glorot_uniform()`

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

**5.1.2.51 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]
```

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

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

**5.1.2.53 hard\_sigmoid()**

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

**5.1.2.54 has\_inf()**

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

**5.1.2.55 has\_nan()**

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

**5.1.2.56 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]
```



**5.1.2.57 identity()**

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

**5.1.2.58 img2col()**

```
auto ceras::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]
```

**5.1.2.59 Input()**

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

**5.1.2.60 is\_valid()**

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

**5.1.2.61 leaky\_relu()**

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

**5.1.2.62 LeakyReLU()**

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

leaky relu activation function.

**5.1.2.63 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]
```

**5.1.2.64 load\_tensor()**

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

**5.1.2.65 log()**

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

**5.1.2.66 mae()**

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

**5.1.2.67 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]
```

#### 5.1.2.68 make\_trainable()

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

Setting an expression's trainable flag

#### 5.1.2.69 max() [1/2]

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

#### 5.1.2.70 max() [2/2]

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

#### 5.1.2.71 max\_pooling\_2d()

```
auto ceras::max_pooling_2d (
    unsigned long stride ) [inline], [noexcept]
```

#### 5.1.2.72 maximum()

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

#### 5.1.2.73 MaxPooling2D()

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

Max pooling operation for 2D spatial data.

**5.1.2.74 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]
```

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

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

**5.1.2.76 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]
```

**5.1.2.77 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 );
```

**5.1.2.78 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]
```

#### 5.1.2.79 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]
```

#### 5.1.2.80 min() [1/2]

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

#### 5.1.2.81 min() [2/2]

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

#### 5.1.2.82 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]
```

#### 5.1.2.83 minus() [2/2]

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

#### 5.1.2.84 mse()

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

#### 5.1.2.85 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 };
```

#### 5.1.2.86 multiply() [1/2]

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

#### 5.1.2.87 multiply() [2/2]

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

#### 5.1.2.88 negative()

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

**5.1.2.89 negative\_relu()**

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

**5.1.2.90 norm()**

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

**5.1.2.91 normalization\_batch()**

```
template<typename T = double>
requires std::floating_point<T> auto ceras::normalization_batch (
    T const momentum = 0.98 ) [inline], [noexcept]
```

**5.1.2.92 ones()**

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

**5.1.2.93 ones\_like() [1/2]**

```
template<Expression Ex>
auto ceras::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.
```

**5.1.2.94 ones\_like() [2/2]**

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

**5.1.2.95 operator"!="()**

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

**5.1.2.96 operator\*() [1/4]**

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

**5.1.2.97 operator\*() [2/4]**

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

**5.1.2.98 operator\*() [3/4]**

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

**5.1.2.99 operator\*() [4/4]**

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



**5.1.2.100 operator+() [1/4]**

```
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+() [2/4]**

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

**5.1.2.102 operator+() [3/4]**

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

**5.1.2.103 operator+() [4/4]**

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

**5.1.2.104 operator-() [1/4]**

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

**5.1.2.105 operator-() [2/4]**

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

**5.1.2.106 operator-() [3/4]**

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

**5.1.2.107 operator-() [4/4]**

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

**5.1.2.108 operator/()**

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

**5.1.2.109 operator<()**

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

**5.1.2.110 operator<<()**

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

**5.1.2.111 operator<=()**

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

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

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

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

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

**5.1.2.114 operator>( )**

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

**5.1.2.115 operator>=( )**

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

**5.1.2.116 plus( )**

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

**5.1.2.117 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} )
```

**5.1.2.118 randn\_like()**

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

**5.1.2.119 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} )
```

**5.1.2.120 random\_like()**

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

**5.1.2.121 random\_normal\_like()**

```
template<typename T = float>
requires std::floating_point<T> auto ceras::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)
```

## 5.1.2.122 read\_tensor()

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

## 5.1.2.123 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]
```

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

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

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

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

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

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

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

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

**5.1.2.128 relu()**

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

**5.1.2.129 ReLU()**

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

Rectified Linear Unit activation function.

**5.1.2.130 repeat()**

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

**5.1.2.131 replace\_placeholder\_with\_expression()**

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

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

## Parameters

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

## Returns

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

**5.1.2.132 repmat()**

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

**5.1.2.133 Reshape()**

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

Reshapes inputs into the given shape.

**5.1.2.134 reshape() [1/2]**

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

**5.1.2.135 reshape() [2/2]**

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

**5.1.2.136 save\_tensor()**

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

**5.1.2.137 selu()**

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

**5.1.2.138 sigmoid()**

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

**5.1.2.139 sign()**

```
template<Expression Ex>
constexpr auto ceras::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);
```

**5.1.2.140 Softmax()**

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

Softmax activation function.



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

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

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

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

**5.1.2.143 softplus()**

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

**5.1.2.144 softsign()**

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

**5.1.2.145 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);
```

#### 5.1.2.146 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]
```

#### 5.1.2.147 squeeze()

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

#### 5.1.2.148 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]
```

#### 5.1.2.149 std()

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

#### 5.1.2.150 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 };
```

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

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

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

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

**5.1.2.153 sum\_reduce()**

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

**5.1.2.154 tanh()**

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

**5.1.2.155 transpose()**

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

**5.1.2.156 truncated\_normal()**

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

**5.1.2.157 up\_sampling\_2d()**

```
auto ceras::up_sampling_2d (
    unsigned long stride ) [inline], [noexcept]
```

**5.1.2.158 update\_cuda\_gemm\_threshold()**

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

**5.1.2.159 UpSampling2D()**

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

Upsampling layer for 2D inputs.

**5.1.2.160 var()**

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

**5.1.2.161 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]
```

**5.1.2.162 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 )
```

**5.1.2.163 zero\_padding\_2d()**

```
auto ceras::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)</tt>.
----------------	--

## 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)

```

## 5.1.2.164 zeros()

```

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

```

## 5.1.2.165 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 Constant

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

### 5.1.3.10 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<>
```

### 5.1.3.11 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 );
        };
    };
}
```

### 5.1.3.12 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`.

### 5.1.3.13 is\_constant\_v

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

### 5.1.3.14 is\_place\_holder\_v

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

### 5.1.3.15 is\_tensor\_v

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

### 5.1.3.16 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`.



#### 5.1.3.17 is\_value\_v

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

#### 5.1.3.18 is\_variable\_v

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

#### 5.1.3.19 MAE

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

##### Initial value:

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

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

#### 5.1.3.20 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;
    };
}
```

#### 5.1.3.21 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;
    };
}
```

### 5.1.3.22 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](#)

### 5.1.3.23 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](#)

### 5.1.3.24 MSE

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

**Initial value:**

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

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

### 5.1.3.25 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<>
```

### 5.1.3.26 Place\_Holder

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

### 5.1.3.27 random\_generator

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

### 5.1.3.28 random\_seed

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

### 5.1.3.29 RMSprop

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

**Initial value:**

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

### 5.1.3.30 SGD

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

**Initial value:**

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

#### 5.1.3.31 Tensor

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

#### 5.1.3.32 Unary\_Operator

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

A type that represents an unary operator.

@concept Unary\_Operator<>

#### 5.1.3.33 Value

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

#### 5.1.3.34 Variable

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

## 5.2 ceras::ceras\_private Namespace Reference

### Classes

- struct [session](#)

## 5.3 ceras::dataset Namespace Reference

### Namespaces

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

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

### Functions

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

## 5.4.1 Function Documentation

### 5.4.1.1 load\_data()

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

Loads the fashion-MNIST dataset.

#### Parameters

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

#### Returns

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

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

#### Example usage:

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

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

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

### Functions

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

## 5.5.1 Function Documentation

### 5.5.1.1 load\_data()

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

Loads the MNIST dataset.

### Parameters

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

### Returns

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

### Example usage:

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

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

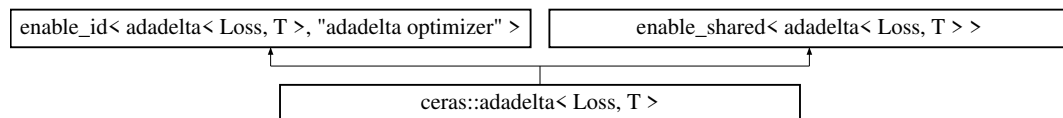
## Chapter 6

# Class Documentation

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

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

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



#### Public Types

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

#### Public Member Functions

- [adadelta](#) (Loss &loss, std::size\_t batch\_size, T rho=0.9) noexcept
- void [forward](#) ()

#### Public Attributes

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

#### 6.1.1 Member Typedef Documentation

### 6.1.1.1 tensor\_type

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

## 6.1.2 Constructor & Destructor Documentation

### 6.1.2.1 adadelta()

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

## 6.1.3 Member Function Documentation

### 6.1.3.1 forward()

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

## 6.1.4 Member Data Documentation

### 6.1.4.1 iterations\_

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

### 6.1.4.2 learning\_rate\_

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



### 6.1.4.3 loss\_

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

### 6.1.4.4 rho\_

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

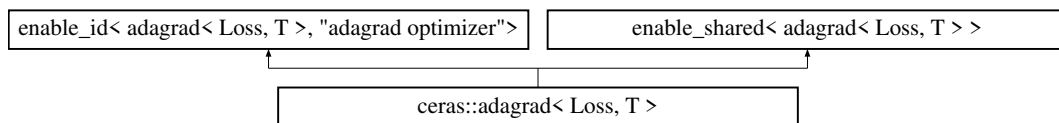
The documentation for this struct was generated from the following file:

- /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_
```

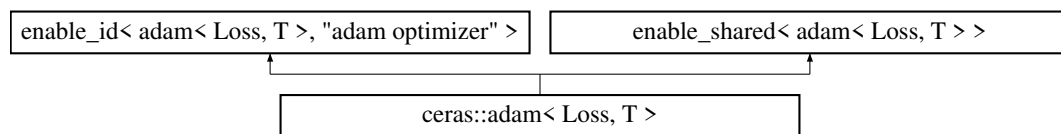
The documentation for this struct was generated from the following file:

- [/data/structured\\_folders/workspace/github.repo/ceras/include/optimizer.hpp](/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_
```

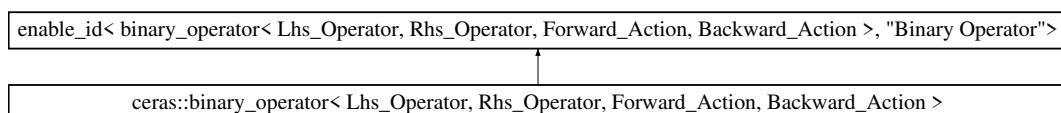
The documentation for this struct was generated from the following file:

- /data/structured\_folders/workspace/github.repo/ceras/include/optimizer.hpp

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

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

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



## Public Types

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

## Public Member Functions

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

## Public Attributes

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

## 6.4.1 Member Typedef Documentation

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

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

## 6.4.2 Constructor & Destructor Documentation

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

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

### 6.4.3 Member Function Documentation

#### 6.4.3.1 backward()

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

#### 6.4.3.2 forward()

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

### 6.4.4 Member Data Documentation

#### 6.4.4.1 backward\_action\_

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

#### 6.4.4.2 forward\_action\_

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

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

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

#### 6.4.4.4 lhs\_op\_

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

#### 6.4.4.5 output\_data\_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
tensor_type ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_↵
Action >::output_data_
```

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

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
tensor_type ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_↵
Action >::rhs_input_data_
```

#### 6.4.4.7 rhs\_op\_

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

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

- [/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
```

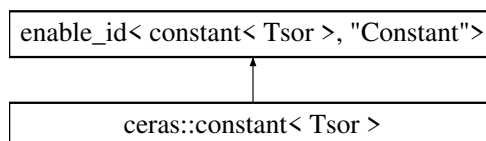
The documentation for this struct was generated from the following file:

- [/data/structured\\_folders/workspace/github.repo/ceras/include/model.hpp](/data/structured_folders/workspace/github.repo/ceras/include/model.hpp)

## 6.6 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\\_](#)

## 6.6.1 Constructor & Destructor Documentation

### 6.6.1.1 constant()

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

## 6.6.2 Member Function Documentation

### 6.6.2.1 backward()

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

### 6.6.2.2 forward()

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

### 6.6.2.3 shape()

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

## 6.6.3 Member Data Documentation

### 6.6.3.1 data\_

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

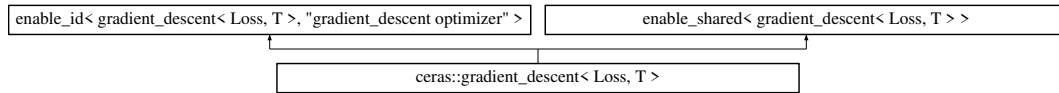
The documentation for this struct was generated from the following file:

- /data/structured\_folders/workspace/github.repo/ceras/include/[constant.hpp](#)

## 6.7 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\\_](#)

## 6.7.1 Member Typedef Documentation

### 6.7.1.1 tensor\_type

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

## 6.7.2 Constructor & Destructor Documentation

### 6.7.2.1 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]
```

## 6.7.3 Member Function Documentation

### 6.7.3.1 forward()

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

## 6.7.4 Member Data Documentation

### 6.7.4.1 learning\_rate\_

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

### 6.7.4.2 loss\_

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

### 6.7.4.3 momentum\_

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

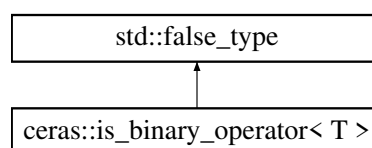
The documentation for this struct was generated from the following file:

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

## 6.8 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:

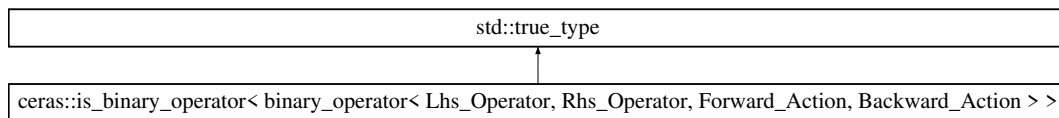
- [/data/structured\\_folders/workspace/github.repo/ceras/include/operation.hpp](/data/structured_folders/workspace/github.repo/ceras/include/operation.hpp)



## 6.9 ceras::is\_binary\_operator< binary\_operator< Lhs\_Operator, Rh\_Operator, Forward\_Action, Backward\_Action > > Struct Template Reference

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

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



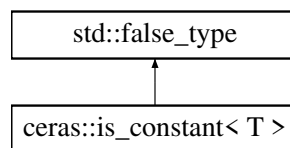
The documentation for this struct was generated from the following file:

- [/data/structured\\_folders/workspace/github.repo/ceras/include/operation.hpp](/data/structured_folders/workspace/github.repo/ceras/include/operation.hpp)

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

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

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



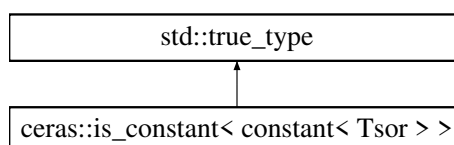
The documentation for this struct was generated from the following file:

- [/data/structured\\_folders/workspace/github.repo/ceras/include/constant.hpp](/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp)

## 6.11 ceras::is\_constant< constant< Ttor > > Struct Template Reference

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

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



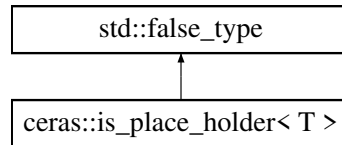
The documentation for this struct was generated from the following file:

- [/data/structured\\_folders/workspace/github.repo/ceras/include/constant.hpp](/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp)

## 6.12 `ceras::is_place_holder< T >` Struct Template Reference

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

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



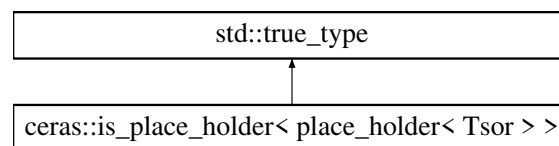
The documentation for this struct was generated from the following file:

- [/data/structured\\_folders/workspace/github.repo/ceras/include/place\\_holder.hpp](/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp)

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

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

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



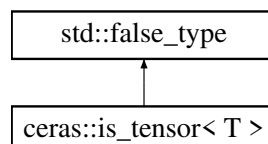
The documentation for this struct was generated from the following file:

- [/data/structured\\_folders/workspace/github.repo/ceras/include/place\\_holder.hpp](/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp)

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

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

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



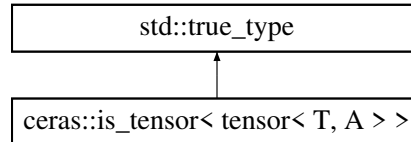
The documentation for this struct was generated from the following file:

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

## 6.15 ceras::is\_tensor< tensor< T, A > > Struct Template Reference

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

Inheritance diagram for ceras::is\_tensor< tensor< T, A > >:



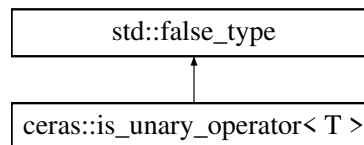
The documentation for this struct was generated from the following file:

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

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

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

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



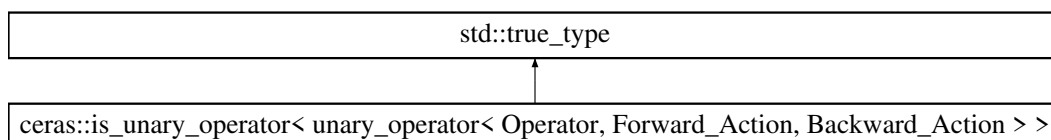
The documentation for this struct was generated from the following file:

- /data/structured\_folders/workspace/github.repo/ceras/include/[operation.hpp](#)

## 6.17 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 > >:



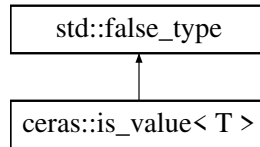
The documentation for this struct was generated from the following file:

- /data/structured\_folders/workspace/github.repo/ceras/include/[operation.hpp](#)

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

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

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



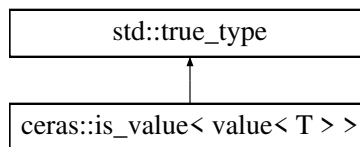
The documentation for this struct was generated from the following file:

- /data/structured\_folders/workspace/github.repo/ceras/include/[value.hpp](#)

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

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

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



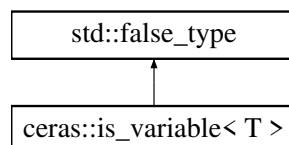
The documentation for this struct was generated from the following file:

- /data/structured\_folders/workspace/github.repo/ceras/include/[value.hpp](#)

## 6.20 ceras::is\_variable< T > Struct Template Reference

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

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



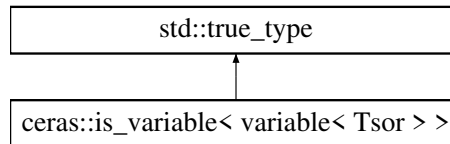
The documentation for this struct was generated from the following file:

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

## 6.21 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](#)

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

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

### Public Types

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

### Public Member Functions

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

### Public Attributes

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

### 6.22.1 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

## 6.22.2 Member Typedef Documentation

### 6.22.2.1 input\_layer\_type

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

### 6.22.2.2 output\_layer\_type

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

## 6.22.3 Constructor & Destructor Documentation

### 6.22.3.1 model()

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

#### Parameters

<a href="#">place_holder</a>	The input layer of the model, a place holder.
<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 };
```

## 6.22.4 Member Function Documentation

### 6.22.4.1 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 usage:

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

### 6.22.4.2 input()

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

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

### 6.22.4.3 load\_weights()

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

Loads all variables from a file

### 6.22.4.4 operator>()

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

Generating a new expression by using the current model.

**Parameters**

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

**Returns**

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

**Example code**

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

**6.22.4.5 output()**

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

Returns the output layer of the model.

**6.22.4.6 predict()**

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

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

**Parameters**

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

**Returns**

The result this model predicts.

**Example to predict**

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



#### 6.22.4.7 save\_weights()

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

Writes all variables to a file

#### 6.22.4.8 summary()

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

Print the model summary to console or to a file.

##### Parameters

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

#### 6.22.4.9 trainable()

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

### 6.22.5 Member Data Documentation

#### 6.22.5.1 expression\_

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

output layer of the model.

#### 6.22.5.2 place\_holder\_

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

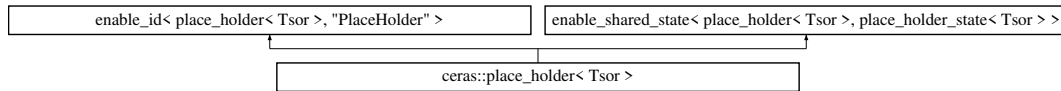
The documentation for this struct was generated from the following file:

- `/data/structured_folders/workspace/github.repo/ceras/include/model.hpp`

## 6.23 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

### 6.23.1 Member Typedef Documentation

#### 6.23.1.1 tensor\_type

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

### 6.23.2 Constructor & Destructor Documentation

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

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

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

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

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

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

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

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

## 6.23.3 Member Function Documentation

### 6.23.3.1 backward()

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

### 6.23.3.2 bind()

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

### 6.23.3.3 forward()

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

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

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

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

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

**6.23.3.6 reset()**

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

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

- [/data/structured\\_folders/workspace/github.repo/ceras/include/place\\_holder.hpp](/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp)

**6.24 ceras::place\_holder\_state< Tsr > Struct Template Reference**

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

**Public Attributes**

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

**6.24.1 Member Data Documentation****6.24.1.1 data\_**

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

### 6.24.1.2 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](#)

## 6.25 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\\_](#)

## 6.25.1 Member Typedef Documentation

### 6.25.1.1 value\_type

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

## 6.25.2 Constructor & Destructor Documentation

### 6.25.2.1 regularizer()

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

## 6.25.3 Member Data Documentation

### 6.25.3.1 l1\_

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

### 6.25.3.2 l2\_

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

### 6.25.3.3 synchronized\_

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

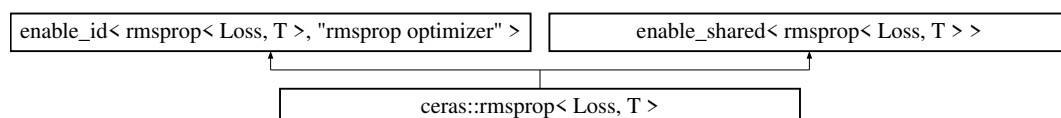
The documentation for this struct was generated from the following file:

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

## 6.26 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\\_](#)

### 6.26.1 Member Typedef Documentation

#### 6.26.1.1 tensor\_type

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

### 6.26.2 Constructor & Destructor Documentation

#### 6.26.2.1 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]
```

### 6.26.3 Member Function Documentation

### 6.26.3.1 forward()

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

## 6.26.4 Member Data Documentation

### 6.26.4.1 decay\_

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

### 6.26.4.2 iterations\_

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

### 6.26.4.3 learning\_rate\_

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

### 6.26.4.4 loss\_

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

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



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

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

### Public Types

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

### Public Member Functions

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

### Public Attributes

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

### 6.27.1 Member Typedef Documentation

#### 6.27.1.1 place\_holder\_type

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

### 6.27.1.2 variable\_state\_type

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

### 6.27.1.3 variable\_type

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

## 6.27.2 Constructor & Destructor Documentation

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

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

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

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

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

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

### 6.27.2.4 ~session()

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

## 6.27.3 Member Function Documentation

### 6.27.3.1 bind()

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

### 6.27.3.2 deserialize()

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

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

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

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

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

### 6.27.3.5 rebind()

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

#### 6.27.3.6 remember()

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

#### 6.27.3.7 restore()

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

#### 6.27.3.8 run()

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

#### 6.27.3.9 save()

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

#### 6.27.3.10 serialize()

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

#### 6.27.3.11 tap()

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

## 6.27.4 Member Data Documentation

### 6.27.4.1 place\_holders\_

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

### 6.27.4.2 variables\_

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

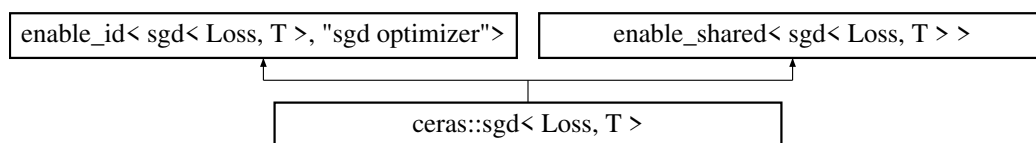
The documentation for this struct was generated from the following file:

- [/data/structured\\_folders/workspace/github.repo/ceras/include/session.hpp](/data/structured_folders/workspace/github.repo/ceras/include/session.hpp)

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

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

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



## Public Types

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

## Public Member Functions

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

## Public Attributes

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

## 6.28.1 Member Typedef Documentation

### 6.28.1.1 `tensor_type`

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

## 6.28.2 Constructor & Destructor Documentation

### 6.28.2.1 `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]
```

## 6.28.3 Member Function Documentation

### 6.28.3.1 `forward()`

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

## 6.28.4 Member Data Documentation

#### 6.28.4.1 decay\_

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

#### 6.28.4.2 iterations\_

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

#### 6.28.4.3 learning\_rate\_

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

#### 6.28.4.4 loss\_

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

#### 6.28.4.5 momentum\_

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

#### 6.28.4.6 nesterov\_

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

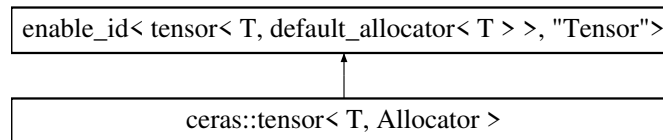
The documentation for this struct was generated from the following file:

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

## 6.29 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

- [tensor slice](#) (unsigned long m, unsigned long n) const noexcept
- 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

## Public Attributes

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

## 6.29.1 Member Typedef Documentation

### 6.29.1.1 allocator

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

### 6.29.1.2 self\_type

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

### 6.29.1.3 shared\_vector

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

### 6.29.1.4 value\_type

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

### 6.29.1.5 vector\_type

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

## 6.29.2 Constructor & Destructor Documentation

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

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

### 6.29.2.2 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]
```

### 6.29.2.3 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]
```

### 6.29.2.4 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]
```

**6.29.2.5 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]
```

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

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

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

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

**6.29.3 Member Function Documentation****6.29.3.1 as\_scalar()**

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

**6.29.3.2 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]
```

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

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

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

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

**6.29.3.5 cbegin()**

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

**6.29.3.6 cend()**

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

**6.29.3.7 copy()**

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

**6.29.3.8 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]
```

**6.29.3.9 data() [1/2]**

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

**6.29.3.10 data() [2/2]**

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

**6.29.3.11 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]
```

**6.29.3.12 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]
```

**6.29.3.13 empty()**

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

**6.29.3.14 end() [1/2]**

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

**6.29.3.15 end() [2/2]**

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

**6.29.3.16 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]
```

**6.29.3.17 ndim()**

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

**6.29.3.18 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]
```

**6.29.3.19 operator\*=( ) [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator*= (
    value_type x ) [inline], [constexpr]
```

**6.29.3.20 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]
```

**6.29.3.21 operator+=( ) [2/2]**

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator+= (
    value_type x ) [inline], [constexpr]
```

**6.29.3.22 operator-()**

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

**6.29.3.23 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]
```

**6.29.3.24 operator-=()** [2/2]

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator-= (
    value_type x ) [inline], [constexpr]
```

**6.29.3.25 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]
```

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

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::operator/= (
    value_type x ) [inline], [constexpr]
```

**6.29.3.27 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]
```

**6.29.3.28 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]
```

**6.29.3.29 operator[]()** [1/2]

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr value_type& ceras::tensor< T, Allocator >::operator[] (
    unsigned long idx ) [inline], [constexpr]
```

**6.29.3.30 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]
```

**6.29.3.31 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();
```

**6.29.3.32 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]
```

**6.29.3.33 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]
```



### 6.29.3.34 shape()

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

### 6.29.3.35 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]
```

### 6.29.3.36 size()

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

### 6.29.3.37 slice()

```
template<typename T , typename Allocator = default_allocator<T>>
tensor ceras::tensor< T, Allocator >::slice (
    unsigned long m,
    unsigned long n ) const [inline], [noexcept]
```

## 6.29.4 Member Data Documentation

### 6.29.4.1 memory\_offset\_

```
template<typename T , typename Allocator = default_allocator<T>>
unsigned long ceras::tensor< T, Allocator >::memory_offset_
```

### 6.29.4.2 shape\_

```
template<typename T , typename Allocator = default_allocator<T>>
std::vector<unsigned long> ceras::tensor< T, Allocator >::shape_
```

### 6.29.4.3 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

## 6.30 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())>

### 6.30.1 Member Typedef Documentation

#### 6.30.1.1 op\_type

```
template<typename L , typename R >
using ceras::tensor_deduction< L, R >::op_type = std::conditional<is_value_v<L>, R, L>::type
```

#### 6.30.1.2 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
```

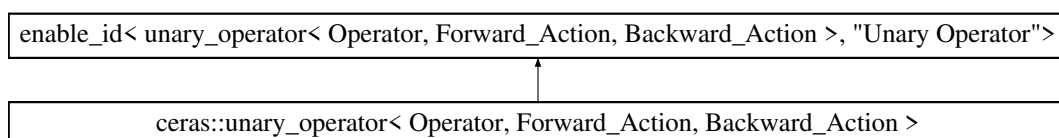
The documentation for this struct was generated from the following file:

- /data/structured\_folders/workspace/github.repo/ceras/include/value.hpp

## 6.31 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\_

### 6.31.1 Constructor & Destructor Documentation

#### 6.31.1.1 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]
```

### 6.31.2 Member Function Documentation

#### 6.31.2.1 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]
```

#### 6.31.2.2 forward()

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
auto ceras::unary_operator< Operator, Forward_Action, Backward_Action >::forward ( ) [inline]
```

### 6.31.3 Member Data Documentation

#### 6.31.3.1 backward\_action\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
Backward_Action ceras::unary_operator< Operator, Forward_Action, Backward_Action >::backward_↵
_action_
```

#### 6.31.3.2 forward\_action\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
Forward_Action ceras::unary_operator< Operator, Forward_Action, Backward_Action >::forward_↵
action_
```

#### 6.31.3.3 input\_data\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
tensor_type ceras::unary_operator< Operator, Forward_Action, Backward_Action >::input_data_
```

#### 6.31.3.4 op\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
Operator ceras::unary_operator< Operator, Forward_Action, Backward_Action >::op_
```

#### 6.31.3.5 output\_data\_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
tensor_type ceras::unary_operator< Operator, Forward_Action, Backward_Action >::output_data_
```

#### 6.31.3.6 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
```

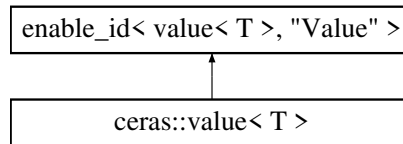
The documentation for this struct was generated from the following file:

- /data/structured\_folders/workspace/github.repo/ceras/include/[operation.hpp](#)

## 6.32 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\_

### 6.32.1 Member Typedef Documentation

#### 6.32.1.1 value\_type

```
template<typename T >
typedef T ceras::value< T >::value_type
```

### 6.32.2 Constructor & Destructor Documentation

**6.32.2.1 value() [1/4]**

```
template<typename T >
ceras::value< T >::value ( ) [delete]
```

**6.32.2.2 value() [2/4]**

```
template<typename T >
ceras::value< T >::value (
    value_type v ) [inline], [noexcept]
```

**6.32.2.3 value() [3/4]**

```
template<typename T >
ceras::value< T >::value (
    value< T > const & ) [default], [noexcept]
```

**6.32.2.4 value() [4/4]**

```
template<typename T >
ceras::value< T >::value (
    value< T > && ) [default], [noexcept]
```

**6.32.3 Member Function Documentation****6.32.3.1 backward()**

```
template<typename T >
void ceras::value< T >::backward (
    auto ) [inline], [noexcept]
```

**6.32.3.2 forward()**

```
template<typename T >
template<Tensor Tsor>
Tsor const ceras::value< T >::forward (
    Tsor const & refer ) const [inline]
```

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

```
template<typename T >
value& ceras::value< T >::operator= (
    value< T > && ) [default], [noexcept]
```

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

```
template<typename T >
value& ceras::value< T >::operator= (
    value< T > const & ) [default], [noexcept]
```

**6.32.4 Member Data Documentation****6.32.4.1 data\_**

```
template<typename T >
value_type ceras::value< T >::data_
```

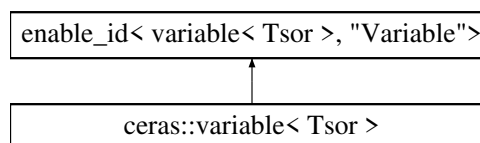
The documentation for this struct was generated from the following file:

- [/data/structured\\_folders/workspace/github.repo/ceras/include/value.hpp](#)

**6.33 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_`

## 6.33.1 Member Typedef Documentation

### 6.33.1.1 `tensor_type`

```
template<Tensor Tsor>
typedef Tsor ceras::variable< Tsor >::tensor_type
```

### 6.33.1.2 `value_type`

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

## 6.33.2 Constructor & Destructor Documentation



**6.33.2.1 variable()** [1/4]

```
template<Tensor Tzor>
ceras::variable< Tzor >::variable (
    tensor_type const & data,
    value_type l1 = value_type{0},
    value_type l2 = value_type{0},
    bool trainable = true ) [inline]
```

**6.33.2.2 variable()** [2/4]

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

**6.33.2.3 variable()** [3/4]

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

**6.33.2.4 variable()** [4/4]

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

**6.33.3 Member Function Documentation****6.33.3.1 backward()**

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

**6.33.3.2 contexts()** [1/2]

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

#### 6.33.3.3 contexts() [2/2]

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

#### 6.33.3.4 data() [1/2]

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

#### 6.33.3.5 data() [2/2]

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

#### 6.33.3.6 forward()

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

#### 6.33.3.7 gradient() [1/2]

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

#### 6.33.3.8 gradient() [2/2]

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

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

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

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

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

#### 6.33.3.11 reset()

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

#### 6.33.3.12 shape()

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

#### 6.33.3.13 trainable() [1/2]

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

#### 6.33.3.14 trainable() [2/2]

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

### 6.33.4 Member Data Documentation

#### 6.33.4.1 regularizer\_

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

#### 6.33.4.2 state\_

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

#### 6.33.4.3 trainable\_

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

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

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

## 6.34 ceras::variable\_state< Tsor > Struct Template Reference

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

### Public Attributes

- Tsor [data\\_](#)
- Tsor [gradient\\_](#)
- std::vector< Tsor > [contexts\\_](#)

### 6.34.1 Member Data Documentation

#### 6.34.1.1 contexts\_

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

#### 6.34.1.2 data\_

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

### 6.34.1.3 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](#)

## 6.35 ceras::view\_2d< T > Struct Template Reference

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

### 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

### Public Attributes

- T \* [data\\_](#)
- unsigned long [row\\_](#)
- unsigned long [col\\_](#)
- bool [transposed\\_](#)

## 6.35.1 Constructor & Destructor Documentation

### 6.35.1.1 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]
```

### 6.35.1.2 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]
```

### 6.35.1.3 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]
```

## 6.35.2 Member Function Documentation

### 6.35.2.1 data() [1/2]

```
template<typename T >
constexpr const T* ceras::view_2d< T >::data ( ) const [inline], [constexpr], [noexcept]
```

### 6.35.2.2 data() [2/2]

```
template<typename T >
constexpr T* ceras::view_2d< T >::data ( ) [inline], [constexpr], [noexcept]
```

### 6.35.2.3 operator[]() [1/2]

```
template<typename T >
constexpr T* ceras::view_2d< T >::operator[] (
    unsigned long index ) [inline], [constexpr]
```

#### 6.35.2.4 operator[]() [2/2]

```
template<typename T >
constexpr const T* ceras::view_2d< T >::operator[] (
    unsigned long index ) const [inline], [constexpr]
```

#### 6.35.2.5 shape()

```
template<typename T >
constexpr auto ceras::view_2d< T >::shape ( ) const [inline], [constexpr], [noexcept]
```

#### 6.35.2.6 size()

```
template<typename T >
constexpr unsigned long ceras::view_2d< T >::size ( ) const [inline], [constexpr], [noexcept]
```

### 6.35.3 Member Data Documentation

#### 6.35.3.1 col\_

```
template<typename T >
unsigned long ceras::view_2d< T >::col_
```

#### 6.35.3.2 data\_

```
template<typename T >
T* ceras::view_2d< T >::data_
```

#### 6.35.3.3 row\_

```
template<typename T >
unsigned long ceras::view_2d< T >::row_
```

### 6.35.3.4 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](#)

## 6.36 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\\_](#)

## 6.36.1 Constructor & Destructor Documentation

### 6.36.1.1 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]
```

## 6.36.2 Member Function Documentation



### 6.36.2.1 operator[]() [1/2]

```
template<typename T >
constexpr auto ceras::view_3d< T >::operator[] (
    unsigned long index ) const [inline], [constexpr], [noexcept]
```

### 6.36.2.2 operator[]() [2/2]

```
template<typename T >
constexpr auto ceras::view_3d< T >::operator[] (
    unsigned long index ) [inline], [constexpr], [noexcept]
```

## 6.36.3 Member Data Documentation

### 6.36.3.1 channel\_

```
template<typename T >
unsigned long ceras::view_3d< T >::channel_
```

### 6.36.3.2 col\_

```
template<typename T >
unsigned long ceras::view_3d< T >::col_
```

### 6.36.3.3 data\_

```
template<typename T >
T* ceras::view_3d< T >::data_
```

### 6.36.3.4 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](#)

## 6.37 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.*

### 6.37.1 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].

### 6.37.2 Constructor & Destructor Documentation

#### 6.37.2.1 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.

### 6.37.3 Member Function Documentation

#### 6.37.3.1 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];
```

#### 6.37.3.2 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;
```

## 6.37.4 Member Data Documentation

### 6.37.4.1 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.

### 6.37.4.2 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.

### 6.37.4.3 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.

### 6.37.4.4 data\_

```
template<typename T >
T* ceras::view_4d< T >::data_
```

The pointer to the start position of the 1-D array.

### 6.37.4.5 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::tanh (Ex const &ex) noexcept`
- `template<Expression Ex>`  
`auto ceras::relu (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`

## 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 "../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/config.hpp File Reference**

## 7.4 `/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>

## 7.5 /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"})

## 7.6 `/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](#)

#### 7.6.1 Macro Definition Documentation



### 7.6.1.1 STB\_IMAGE\_IMPLEMENTATION

```
#define STB_IMAGE_IMPLEMENTATION
```

### 7.6.1.2 STB\_IMAGE\_RESIZE\_IMPLEMENTATION

```
#define STB_IMAGE_RESIZE_IMPLEMENTATION
```

### 7.6.1.3 STB\_IMAGE\_WRITE\_IMPLEMENTATION

```
#define STB_IMAGE_WRITE_IMPLEMENTATION
```

## 7.7 /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) noex-  
cept
- 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

## 7.8 /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 Rhc\_Expression>  
constexpr auto [ceras::mean\\_squared\\_logarithmic\\_error](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression  
const &rhc\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::squared\\_loss](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex) noex-  
cept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::mean\\_squared\\_error](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex)  
noexcept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::mse](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhc\_Expression>  
constexpr auto [ceras::abs\\_loss](#) (Lhs\_Expression const &lhc\_ex, Rhc\_Expression const &rhc\_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`

## 7.9 /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)`

## 7.10 [/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 Rhs\_Expression>  
constexpr auto [ceras::plus](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
constexpr auto [ceras::operator+](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
auto [ceras::operator\\*](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::log](#) (Ex const &ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::negative](#) (Ex const &ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
constexpr auto [ceras::elementwise\\_product](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
constexpr auto [ceras::elementwise\\_multiply](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
constexpr auto [ceras::hadamard\\_product](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_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 Rhs\_Expression>  
constexpr auto [ceras::minus](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
constexpr auto [ceras::operator-](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::square](#) (Ex const &ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::abs](#) (Ex const &ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::exp](#) (Ex const &ex) noexcept
- template<typename Float >  
requires std::floating\_point< Float > constexpr auto [ceras::clip](#) (Float lower, Float upper=std::numeric\_limits< Float >::max()) noexcept
- auto [ceras::reshape](#) (std::vector< unsigned long > const &new\_shape, bool include\_batch\_flag=true) noexcept
- template<Expression Ex>  
constexpr auto [ceras::flatten](#) (Ex const &ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::identity](#) (Ex const &ex) noexcept
- template<Expression Ex>  
auto [ceras::transpose](#) (Ex const &ex) noexcept
- auto [ceras::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) noexcept

- auto [ceras::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 [ceras::drop\\_out](#) (T const factor) noexcept
- auto [ceras::max\\_pooling\\_2d](#) (unsigned long stride) noexcept
- auto [ceras::average\\_pooling\\_2d](#) (unsigned long stride) noexcept
- auto [ceras::up\\_sampling\\_2d](#) (unsigned long stride) noexcept
- template<typename T = double>  
requires std::floating\_point< T > auto [ceras::normalization\\_batch](#) (T const momentum=0.98) noexcept
- template<typename T >  
requires std::floating\_point< T > auto [ceras::batch\\_normalization](#) (T const momentum=0.98) noexcept
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
constexpr auto [ceras::concatenate](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- auto [ceras::concatenate](#) (unsigned long axe=-1)
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
constexpr auto [ceras::concat](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- auto [ceras::concat](#) (unsigned long axe=-1)
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
constexpr auto [ceras::maximum](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- template<typename T = float>  
requires std::floating\_point< T > auto [ceras::random\\_normal\\_like](#) (T mean=0.0, T stddev=1.0) noexcept
- template<Expression Ex>  
auto [ceras::ones\\_like](#) (Ex const &ex) noexcept
- template<Expression Lhs\_Expression, Expression Rhs\_Expression>  
constexpr auto [ceras::equal](#) (Lhs\_Expression const &lhs\_ex, Rhs\_Expression const &rhs\_ex) noexcept
- template<Expression Ex>  
constexpr auto [ceras::sign](#) (Ex const &ex) noexcept
- auto [ceras::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).*

## 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.*

## 7.11 /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](#) = adadelta< Loss, T >

### Variables

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

## 7.12 `/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>`



## 7.13 /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< Tzor >](#)

### Namespaces

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

### Functions

- template<Tensor Tzor>  
ceras\_private::session< Tzor > & [ceras::get\\_default\\_session](#) ()

## 7.14 /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< 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 Tsr, typename CharT , typename Traits >`  
`std::basic_ostream< CharT, Traits >` & [ceras::operator<<](#) (`std::basic_ostream< CharT, Traits >` &os\_, Tsr const &tsr)
- `template<typename T >`  
requires `std::floating_point< T >` void [ceras::gemm\\_cpu](#) (T const \*A, bool a\_transposed, T const \*B, bool b\_transposed, unsigned long m, unsigned long n, unsigned long k, T \*C)
- void [ceras::update\\_cuda\\_gemm\\_threshold](#) ()
- `template<typename T >`  
requires `std::floating_point< T >` void [ceras::gemm](#) (T const \*A, bool a\_transposed, T const \*B, bool b\_↔ transposed, unsigned long m, unsigned long n, unsigned long k, T \*C)
- `template<typename T >`  
requires `std::floating_point< T >` void [ceras::gemm](#) (`view_2d< T >` const &x, `view_2d< T >` const &y, `view_2d< T >` &ans)
- `template<Tensor Tsr>`  
Tsr [ceras::add](#) (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::operator+](#) (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::operator+](#) (typename Tsr::value\_type const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::operator+](#) (Tsr const &lhs, typename Tsr::value\_type const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::minus](#) (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::operator-](#) (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::operator-](#) (typename Tsr::value\_type const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::operator-](#) (Tsr const &lhs, typename Tsr::value\_type const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::operator\\*](#) (typename Tsr::value\_type const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::operator\\*](#) (Tsr const &lhs, typename Tsr::value\_type const &rhs) noexcept
- `template<Tensor Tsr>`  
Tsr [ceras::operator/](#) (Tsr const &lhs, typename Tsr::value\_type const &rhs) noexcept

- `template<Tensor 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 Tsor>`  
`Tsor ceras::random\_like (Tsor const &tsor, typename Tsor::value_type min=0, typename Tsor::value_type max=1)`
- `template<Tensor Tsor>`  
`Tsor ceras::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 > ceras::glorot\_uniform (std::initializer_list< unsigned long > shape)`
- `template<Tensor Tsor>`  
`Tsor ceras::deep\_copy (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`Tsor ceras::copy (Tsor const &tsor)`
- `template<Tensor Tsor>`  
`Tsor ceras::concatenate (Tsor const &lhs, Tsor const &rhs, unsigned long axis=0) noexcept`
- `template<Tensor Tsor>`  
`Tsor ceras::repmat (Tsor const &tsor, unsigned long row_rep, unsigned long col_rep)`
- `template<Tensor Tsor>`  
`constexpr bool ceras::empty (Tsor 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 Tsor>`  
`constexpr Tsor ceras::zeros\_like (Tsor const &tsor)`
- `template<typename T , typename A = default_allocator<T>>`  
`constexpr tensor< T, A > ceras::ones (std::vector< unsigned long > const &shape)`
- `template<Tensor Tsor>`  
`constexpr Tsor ceras::ones\_like (Tsor const &tsor)`

- `template<Tensor Tsr>`  
`auto ceras::max (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::amax (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::min (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::amin (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::sum (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::mean (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`auto ceras::norm (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`Tsr ceras::abs (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`Tsr ceras::softmax (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`bool ceras::has_nan (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`bool ceras::has_inf (Tsr const &tsr)`
- `template<Tensor Tsr>`  
`bool ceras::is_valid (Tsr const &tsr)`
- `template<Tensor Tsr, typename Function >`  
`Tsr ceras::reduce (Tsr const &ts, unsigned long axis, typename Tsr::value_type const &init, Function const &func, bool keepdims=false) noexcept`
- `template<Tensor Tsr>`  
`Tsr ceras::sum (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsr>`  
`requires std::floating_point< typename Tsr::value_type > Tsr ceras::mean (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsr>`  
`requires std::floating_point< typename Tsr::value_type > Tsr ceras::variance (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsr>`  
`requires std::floating_point< typename Tsr::value_type > Tsr ceras::standard_deviation (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsr>`  
`requires std::floating_point< typename Tsr::value_type > Tsr::value_type ceras::var (Tsr const &ts) noexcept`
- `template<Tensor Tsr>`  
`requires std::floating_point< typename Tsr::value_type > Tsr::value_type ceras::std (Tsr const &ts) noexcept`
- `template<Tensor Tsr>`  
`Tsr ceras::max (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsr>`  
`Tsr ceras::min (Tsr const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<typename T, typename A = default_allocator<T>>`  
`requires std::floating_point< T > tensor< T, A > ceras::linspace (T start, T stop, unsigned long num, bool endpoint=true) noexcept`
- `template<class _Tp, class _CharT, class _Traits, class _Alloc >`  
`std::basic_istream< _CharT, _Traits > & ceras::read_tensor (std::basic_istream< _CharT, _Traits > &__is, tensor< _Tp, _Alloc > &__x)`
- `template<class _Tp, class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & ceras::write_tensor (std::basic_ostream< _CharT, _Traits > &__os, tensor< _Tp, _Alloc > const &__x)`

- `template<typename T, typename A = default_allocator<T>>`  
`tensor< T, A > ceras::load\_tensor (std::string const &file_name)`
- `template<Tensor 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>`

## 7.15 /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>`

## 7.16 `/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< Tsor >](#)
- struct [ceras::regularizer< Float >](#)
- struct [ceras::variable< Tsor >](#)
- struct [ceras::is\\_variable< T >](#)
- struct [ceras::is\\_variable< variable< Tsor > >](#)

### Namespaces

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

### Functions

- `template<Tensor Tsor>`  
[ceras\\_private::session< Tsor >](#) & [ceras::get\\_default\\_session](#) ()
- `template<Variable Var>`  
 bool [ceras::operator==](#) (Var const &lhs, Var const &rhs) noexcept

### Variables

- `template<class T >`  
 constexpr bool [ceras::is\\_variable\\_v](#) = [is\\_variable<T>::value](#)
- `template<typename T >`  
 concept [ceras::Variable](#) = [is\\_variable\\_v<T>](#)

# Index

/data/structured\_folders/workspace/github.repo/ceras/include/activation\_ops.h 64  
131 Adam  
/data/structured\_folders/workspace/github.repo/ceras/include/ceras.h 62  
132 adam  
/data/structured\_folders/workspace/github.repo/ceras/include/config\_ops.h 66  
132 Add  
/data/structured\_folders/workspace/github.repo/ceras/include/constant\_ops.h 19  
132 add  
/data/structured\_folders/workspace/github.repo/ceras/include/data\_ops.h 19  
133 allocator  
/data/structured\_folders/workspace/github.repo/ceras/include/include\_ops.h 103  
134 amax  
/data/structured\_folders/workspace/github.repo/ceras/include/layer\_ops.h 19  
135 amin  
/data/structured\_folders/workspace/github.repo/ceras/include/loss\_ops.h 20  
136 amsgrad\_  
/data/structured\_folders/workspace/github.repo/ceras/include/model\_ops.h 66  
137 as\_scalar  
/data/structured\_folders/workspace/github.repo/ceras/include/operation\_ops.h 105  
138 as\_tensor  
/data/structured\_folders/workspace/github.repo/ceras/include/operation\_ops.h 20  
141 as\_type  
/data/structured\_folders/workspace/github.repo/ceras/include/place\_holder\_ops.h 105  
142 average\_pooling\_2d  
/data/structured\_folders/workspace/github.repo/ceras/include/session\_ops.h 19  
143 AveragePooling2D  
/data/structured\_folders/workspace/github.repo/ceras/include/tensor\_ops.h 20  
143  
/data/structured\_folders/workspace/github.repo/ceras/include/value\_ops.h  
147 ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator,  
/data/structured\_folders/workspace/github.repo/ceras/include/variable\_ops.h  
148 Forward\_Action, Backward\_Action >, 69  
ceras::constant< T>, 76  
ceras::place\_holder< T>, 89  
ceras::unary\_operator< Operator, Forward\_Action,  
Backward\_Action >, 113  
ceras::value< T >, 116  
ceras::variable< T>, 119  
backward\_action\_  
ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator,  
Forward\_Action, Backward\_Action >, 69  
ceras::unary\_operator< Operator, Forward\_Action,  
Backward\_Action >, 114  
batch\_normalization  
ceras, 20  
batch\_size\_  
ceras::view\_4d< T >, 130  
BatchNormalization  
ceras, 20, 21  
begin  
ceras::tensor< T, Allocator >, 105

/data/structured\_folders/workspace/github.repo/ceras/include/activation\_ops.h 64  
131 Adam  
/data/structured\_folders/workspace/github.repo/ceras/include/ceras.h 62  
132 adam  
/data/structured\_folders/workspace/github.repo/ceras/include/config\_ops.h 66  
132 Add  
/data/structured\_folders/workspace/github.repo/ceras/include/constant\_ops.h 19  
132 add  
/data/structured\_folders/workspace/github.repo/ceras/include/data\_ops.h 19  
133 allocator  
/data/structured\_folders/workspace/github.repo/ceras/include/include\_ops.h 103  
134 amax  
/data/structured\_folders/workspace/github.repo/ceras/include/layer\_ops.h 19  
135 amin  
/data/structured\_folders/workspace/github.repo/ceras/include/loss\_ops.h 20  
136 amsgrad\_  
/data/structured\_folders/workspace/github.repo/ceras/include/model\_ops.h 66  
137 as\_scalar  
/data/structured\_folders/workspace/github.repo/ceras/include/operation\_ops.h 105  
138 as\_tensor  
/data/structured\_folders/workspace/github.repo/ceras/include/operation\_ops.h 20  
141 as\_type  
/data/structured\_folders/workspace/github.repo/ceras/include/place\_holder\_ops.h 105  
142 average\_pooling\_2d  
/data/structured\_folders/workspace/github.repo/ceras/include/session\_ops.h 19  
143 AveragePooling2D  
/data/structured\_folders/workspace/github.repo/ceras/include/tensor\_ops.h 20  
143  
/data/structured\_folders/workspace/github.repo/ceras/include/value\_ops.h  
147 ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator,  
/data/structured\_folders/workspace/github.repo/ceras/include/variable\_ops.h  
148 Forward\_Action, Backward\_Action >, 69  
ceras::constant< T>, 76  
ceras::place\_holder< T>, 89  
ceras::unary\_operator< Operator, Forward\_Action,  
Backward\_Action >, 113  
ceras::value< T >, 116  
ceras::variable< T>, 119  
backward\_action\_  
ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator,  
Forward\_Action, Backward\_Action >, 69  
ceras::unary\_operator< Operator, Forward\_Action,  
Backward\_Action >, 114  
batch\_normalization  
ceras, 20  
batch\_size\_  
ceras::view\_4d< T >, 130  
BatchNormalization  
ceras, 20, 21  
begin  
ceras::tensor< T, Allocator >, 105

~session  
ceras::ceras\_private::session< T>, 96

abs  
ceras, 19  
abs\_loss  
ceras, 19  
ada\_delta  
ceras, 17  
ada\_grad  
ceras, 18  
Adadelta  
ceras, 51  
adadelta  
ceras::adadelta< Loss, T >, 62  
Adagrad  
ceras, 51  
adagrad

- beta\_1\_
  - ceras::adam< Loss, T >, 66
- beta\_2\_
  - ceras::adam< Loss, T >, 67
- binary\_cross\_entropy\_loss
  - ceras, 21
- Binary\_Operator
  - ceras, 52
- binary\_operator
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 68
- BinaryCrossEntropy
  - ceras, 52
- BinaryCrossentropy
  - ceras, 52
- bind
  - ceras::ceras\_private::session< Tsor >, 97
  - ceras::place\_holder< Tsor >, 89
- CategoricalCrossEntropy
  - ceras, 53
- CategoricalCrossentropy
  - ceras, 53
- cbegin
  - ceras::tensor< T, Allocator >, 106
- cend
  - ceras::tensor< T, Allocator >, 106
- ceras, 9
  - abs, 19
  - abs\_loss, 19
  - ada\_delta, 17
  - ada\_grad, 18
  - Adadelta, 51
  - Adagrad, 51
  - Adam, 52
  - Add, 19
  - add, 19
  - amax, 19
  - amin, 20
  - as\_tensor, 20
  - average\_pooling\_2d, 20
  - AveragePooling2D, 20
  - batch\_normalization, 20
  - BatchNormalization, 20, 21
  - binary\_cross\_entropy\_loss, 21
  - Binary\_Operator, 52
  - BinaryCrossEntropy, 52
  - BinaryCrossentropy, 52
  - CategoricalCrossEntropy, 53
  - CategoricalCrossentropy, 53
  - clip, 21, 22
  - computation\_graph, 22
  - concat, 22
  - Concatenate, 23
  - concatenate, 22, 23
  - Constant, 53
  - Conv2D, 23
  - conv2d, 24
  - copy, 24
  - cross\_entropy, 24
  - cross\_entropy\_loss, 25
  - cube, 18
  - deep\_copy, 25
  - default\_allocator, 18
  - Dense, 25
  - drop\_out, 26
  - Dropout, 26
  - elementwise\_divide, 26
  - elementwise\_multiply, 26
  - elementwise\_product, 26
  - ELU, 27
  - elu, 27
  - empty, 27
  - equal, 27
  - exp, 28
  - exponential, 28
  - Expression, 53
  - Flatten, 28
  - flatten, 28
  - gelu, 28
  - gemm, 28, 29
  - gemm\_cpu, 29
  - get\_default\_session, 29
  - glorot\_uniform, 29
  - hadamard\_product, 29, 30
  - hard\_sigmoid, 30
  - has\_inf, 30
  - has\_nan, 30
  - Hinge, 53
  - hinge\_loss, 30
  - identity, 30
  - img2col, 31
  - Input, 31
  - is\_binary\_operator\_v, 54
  - is\_constant\_v, 54
  - is\_place\_holder\_v, 54
  - is\_tensor\_v, 54
  - is\_unary\_operator\_v, 54
  - is\_valid, 31
  - is\_value\_v, 54
  - is\_variable\_v, 55
  - leaky\_relu, 31
  - LeakyReLU, 31
  - linspace, 31
  - load\_tensor, 32
  - log, 32
  - MAE, 55
  - mae, 32
  - make\_binary\_operator, 55
  - make\_compiled\_model, 32
  - make\_trainable, 32
  - make\_unary\_operator, 55
  - matrix, 18
  - max, 33
  - max\_pooling\_2d, 33
  - maximum, 33
  - MaxPooling2D, 33



- mean, [33](#), [34](#)
- mean\_absolute\_error, [34](#)
- mean\_reduce, [34](#)
- mean\_squared\_error, [34](#)
- mean\_squared\_logarithmic\_error, [35](#)
- MeanAbsoluteError, [55](#)
- MeanSquaredError, [56](#)
- min, [35](#)
- minus, [35](#)
- MSE, [56](#)
- mse, [35](#)
- Multiply, [36](#)
- multiply, [36](#)
- negative, [36](#)
- negative\_relu, [36](#)
- norm, [37](#)
- normalization\_batch, [37](#)
- ones, [37](#)
- ones\_like, [37](#)
- Operator, [56](#)
- operator!=, [38](#)
- operator<, [40](#)
- operator<<, [40](#)
- operator<=, [40](#)
- operator>, [41](#)
- operator>=, [41](#)
- operator\*, [38](#)
- operator+, [38](#), [39](#)
- operator-, [39](#), [40](#)
- operator/, [40](#)
- operator==, [40](#), [41](#)
- Place\_Holder, [57](#)
- plus, [41](#)
- randn, [41](#)
- randn\_like, [42](#)
- random, [42](#)
- random\_generator, [57](#)
- random\_like, [42](#)
- random\_normal\_like, [42](#)
- random\_seed, [57](#)
- read\_tensor, [43](#)
- reduce, [43](#)
- reduce\_mean, [43](#)
- reduce\_sum, [43](#), [44](#)
- ReLU, [44](#)
- relu, [44](#)
- repeat, [44](#)
- replace\_placeholder\_with\_expression, [44](#)
- repmat, [45](#)
- Reshape, [45](#)
- reshape, [45](#)
- rms\_prop, [18](#)
- RMSprop, [57](#)
- save\_tensor, [45](#)
- selu, [46](#)
- SGD, [57](#)
- sigmoid, [46](#)
- sign, [46](#)
- Softmax, [46](#)
- softmax, [46](#), [47](#)
- softplus, [47](#)
- softsign, [47](#)
- square, [47](#)
- squared\_loss, [48](#)
- squeeze, [48](#)
- standard\_deviation, [48](#)
- std, [48](#)
- Subtract, [48](#)
- sum, [48](#), [49](#)
- sum\_reduce, [49](#)
- tanh, [49](#)
- Tensor, [57](#)
- tesseract, [18](#)
- transpose, [49](#)
- truncated\_normal, [49](#)
- Unary\_Operator, [58](#)
- up\_sampling\_2d, [49](#)
- update\_cuda\_gemm\_threshold, [50](#)
- UpSampling2D, [50](#)
- Value, [58](#)
- var, [50](#)
- Variable, [58](#)
- variance, [50](#)
- write\_tensor, [50](#)
- zero\_padding\_2d, [50](#)
- zeros, [51](#)
- zeros\_like, [51](#)
- ceras::adadelta< Loss, T >, [61](#)
  - adadelta, [62](#)
  - forward, [62](#)
  - iterations\_, [62](#)
  - learning\_rate\_, [62](#)
  - loss\_, [62](#)
  - rho\_, [63](#)
  - tensor\_type, [61](#)
- ceras::adagrad< Loss, T >, [63](#)
  - adagrad, [64](#)
  - decay\_, [64](#)
  - forward, [64](#)
  - iterations\_, [64](#)
  - learning\_rate\_, [64](#)
  - loss\_, [65](#)
  - tensor\_type, [63](#)
- ceras::adam< Loss, T >, [65](#)
  - adam, [66](#)
  - amsgrad\_, [66](#)
  - beta\_1\_, [66](#)
  - beta\_2\_, [67](#)
  - forward, [66](#)
  - iterations\_, [67](#)
  - learning\_rate\_, [67](#)
  - loss\_, [67](#)
  - tensor\_type, [66](#)
- ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, [67](#)
  - backward, [69](#)

- backward\_action\_, 69
- binary\_operator, 68
- forward, 69
- forward\_action\_, 69
- lhs\_input\_data\_, 69
- lhs\_op\_, 69
- output\_data\_, 70
- rhs\_input\_data\_, 70
- rhs\_op\_, 70
- tensor\_type, 68
- ceras::ceras\_private, 58
- ceras::ceras\_private::session< Tzor >, 95
  - ~session, 96
  - bind, 97
  - deserialize, 97
  - operator=, 97
  - place\_holder\_type, 95
  - place\_holders\_, 99
  - rebind, 97
  - remember, 97
  - restore, 98
  - run, 98
  - save, 98
  - serialize, 98
  - session, 96
  - tap, 98
  - variable\_state\_type, 95
  - variable\_type, 96
  - variables\_, 99
- ceras::compiled\_model< Model, Optimizer, Loss >, 70
  - compiled\_model, 71
  - compiled\_optimizer\_, 74
  - evaluate, 72
  - fit, 72
  - ground\_truth\_place\_holder\_, 74
  - input\_place\_holder\_, 74
  - io\_layer\_type, 71
  - loss\_, 74
  - model\_, 74
  - operator(), 73
  - optimizer\_, 75
  - optimizer\_type, 75
  - predict, 73
  - train\_on\_batch, 73
  - trainable, 74
- ceras::constant< Tzor >, 75
  - backward, 76
  - constant, 76
  - data\_, 76
  - forward, 76
  - shape, 76
- ceras::dataset, 58
- ceras::dataset::fashion\_mnist, 58
  - load\_data, 59
- ceras::dataset::mnist, 59
  - load\_data, 59
- ceras::gradient\_descent< Loss, T >, 77
  - forward, 78
  - gradient\_descent, 77
  - learning\_rate\_, 78
  - loss\_, 78
  - momentum\_, 78
  - tensor\_type, 77
- ceras::is\_binary\_operator< binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action > >, 79
- ceras::is\_binary\_operator< T >, 78
- ceras::is\_constant< constant< Tzor > >, 79
- ceras::is\_constant< T >, 79
- ceras::is\_place\_holder< place\_holder< Tzor > >, 80
- ceras::is\_place\_holder< T >, 80
- ceras::is\_tensor< T >, 80
- ceras::is\_tensor< tensor< T, A > >, 81
- ceras::is\_unary\_operator< T >, 81
- ceras::is\_unary\_operator< unary\_operator< Operator, Forward\_Action, Backward\_Action > >, 81
- ceras::is\_value< T >, 82
- ceras::is\_value< value< T > >, 82
- ceras::is\_variable< T >, 82
- ceras::is\_variable< variable< Tzor > >, 83
- ceras::model< Ex, Ph >, 83
  - compile, 84
  - expression\_, 87
  - input, 85
  - input\_layer\_type, 84
  - load\_weights, 85
  - model, 84
  - operator(), 85
  - output, 86
  - output\_layer\_type, 84
  - place\_holder\_, 87
  - predict, 86
  - save\_weights, 86
  - summary, 87
  - trainable, 87
- ceras::place\_holder< Tzor >, 88
  - backward, 89
  - bind, 89
  - forward, 89
  - operator=, 89, 90
  - place\_holder, 88, 89
  - reset, 90
  - tensor\_type, 88
- ceras::place\_holder\_state< Tzor >, 90
  - data\_, 90
  - shape\_hint\_, 90
- ceras::regularizer< Float >, 91
  - l1\_, 92
  - l2\_, 92
  - regularizer, 91
  - synchronized\_, 92
  - value\_type, 91
- ceras::rmsprop< Loss, T >, 92
  - decay\_, 94
  - forward, 93
  - iterations\_, 94

- learning\_rate\_, 94
- loss\_, 94
- rho\_, 94
- rmsprop, 93
- tensor\_type, 93
- ceras::sgd< Loss, T >, 99
  - decay\_, 100
  - forward, 100
  - iterations\_, 101
  - learning\_rate\_, 101
  - loss\_, 101
  - momentum\_, 101
  - nesterov\_, 101
  - sgd, 100
  - tensor\_type, 100
- ceras::tensor< T, Allocator >, 102
  - allocator, 103
  - as\_scalar, 105
  - as\_type, 105
  - begin, 105
  - cbegin, 106
  - cend, 106
  - copy, 106
  - creep\_to, 106
  - data, 106
  - deep\_copy, 106, 107
  - empty, 107
  - end, 107
  - map, 107
  - memory\_offset\_, 111
  - ndim, 107
  - operator\*=: 108
  - operator+=, 108
  - operator-, 108
  - operator-=, 108, 109
  - operator/=: 109
  - operator=, 109
  - operator[], 109, 110
  - reset, 110
  - reshape, 110
  - resize, 110
  - self\_type, 103
  - shape, 110
  - shape\_, 111
  - shared\_vector, 103
  - shrink\_to, 111
  - size, 111
  - slice, 111
  - tensor, 104, 105
  - value\_type, 103
  - vector\_, 111
  - vector\_type, 104
- ceras::tensor\_deduction< L, R >, 112
  - op\_type, 112
  - tensor\_type, 112
- ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 112
  - backward, 113
  - backward\_action\_, 114
  - forward, 113
  - forward\_action\_, 114
  - input\_data\_, 114
  - op\_, 114
  - output\_data\_, 114
  - tensor\_type, 114
  - unary\_operator, 113
- ceras::value< T >, 115
  - backward, 116
  - data\_, 117
  - forward, 116
  - operator=, 116, 117
  - value, 115, 116
  - value\_type, 115
- ceras::variable< Tsor >, 117
  - backward, 119
  - contexts, 119
  - data, 120
  - forward, 120
  - gradient, 120
  - operator=, 120
  - regularizer\_, 121
  - reset, 121
  - shape, 121
  - state\_, 121
  - tensor\_type, 118
  - trainable, 121
  - trainable\_, 122
  - value\_type, 118
  - variable, 118, 119
- ceras::variable\_state< Tsor >, 122
  - contexts\_, 122
  - data\_, 122
  - gradient\_, 122
- ceras::view\_2d< T >, 123
  - col\_, 125
  - data, 124
  - data\_, 125
  - operator[], 124
  - row\_, 125
  - shape, 125
  - size, 125
  - transposed\_, 125
  - view\_2d, 123, 124
- ceras::view\_3d< T >, 126
  - channel\_, 127
  - col\_, 127
  - data\_, 127
  - operator[], 126, 127
  - row\_, 127
  - view\_3d, 126
- ceras::view\_4d< T >, 128
  - batch\_size\_, 130
  - channel\_, 130
  - col\_, 130
  - data\_, 130
  - operator[], 129

- row\_, 130
- view\_4d, 128
- channel\_
  - ceras::view\_3d< T >, 127
  - ceras::view\_4d< T >, 130
- clip
  - ceras, 21, 22
- col\_
  - ceras::view\_2d< T >, 125
  - ceras::view\_3d< T >, 127
  - ceras::view\_4d< T >, 130
- compile
  - ceras::model< Ex, Ph >, 84
- compiled\_model
  - ceras::compiled\_model< Model, Optimizer, Loss >, 71
- compiled\_optimizer\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, 74
- computation\_graph
  - ceras, 22
- concat
  - ceras, 22
- Concatenate
  - ceras, 23
- concatenate
  - ceras, 22, 23
- Constant
  - ceras, 53
- constant
  - ceras::constant< T, Allocator >, 76
- contexts
  - ceras::variable< T, Allocator >, 119
- contexts\_
  - ceras::variable\_state< T, Allocator >, 122
- Conv2D
  - ceras, 23
- conv2d
  - ceras, 24
- copy
  - ceras, 24
  - ceras::tensor< T, Allocator >, 106
- creep\_to
  - ceras::tensor< T, Allocator >, 106
- cross\_entropy
  - ceras, 24
- cross\_entropy\_loss
  - ceras, 25
- cube
  - ceras, 18
- data
  - ceras::tensor< T, Allocator >, 106
  - ceras::variable< T, Allocator >, 120
  - ceras::view\_2d< T >, 124
- data\_
  - ceras::constant< T, Allocator >, 76
  - ceras::place\_holder\_state< T, Allocator >, 90
  - ceras::value< T >, 117
- ceras::variable\_state< T, Allocator >, 122
- ceras::view\_2d< T >, 125
- ceras::view\_3d< T >, 127
- ceras::view\_4d< T >, 130
- decay\_
  - ceras::adagrad< Loss, T >, 64
  - ceras::rmsprop< Loss, T >, 94
  - ceras::sgd< Loss, T >, 100
- deep\_copy
  - ceras, 25
  - ceras::tensor< T, Allocator >, 106, 107
- default\_allocator
  - ceras, 18
- Dense
  - ceras, 25
- deserialize
  - ceras::ceras\_private::session< T, Allocator >, 97
- drop\_out
  - ceras, 26
- Dropout
  - ceras, 26
- elementwise\_divide
  - ceras, 26
- elementwise\_multiply
  - ceras, 26
- elementwise\_product
  - ceras, 26
- ELU
  - ceras, 27
- elu
  - ceras, 27
- empty
  - ceras, 27
  - ceras::tensor< T, Allocator >, 107
- end
  - ceras::tensor< T, Allocator >, 107
- equal
  - ceras, 27
- evaluate
  - ceras::compiled\_model< Model, Optimizer, Loss >, 72
- exp
  - ceras, 28
- exponential
  - ceras, 28
- Expression
  - ceras, 53
- expression\_
  - ceras::model< Ex, Ph >, 87
- fit
  - ceras::compiled\_model< Model, Optimizer, Loss >, 72
- Flatten
  - ceras, 28
- flatten
  - ceras, 28
- forward

- ceras::adadelta< Loss, T >, 62
  - ceras::adagrad< Loss, T >, 64
  - ceras::adam< Loss, T >, 66
  - ceras::binary\_operator< Lhs\_Operator, Rh\_Operator, Forward\_Action, Backward\_Action >, 69
  - ceras::constant< T\_Operator >, 76
  - ceras::gradient\_descent< Loss, T >, 78
  - ceras::place\_holder< T\_Operator >, 89
  - ceras::rmsprop< Loss, T >, 93
  - ceras::sgd< Loss, T >, 100
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 113
  - ceras::value< T >, 116
  - ceras::variable< T\_Operator >, 120
- forward\_action\_
  - ceras::binary\_operator< Lhs\_Operator, Rh\_Operator, Forward\_Action, Backward\_Action >, 69
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 114
- gelu
  - ceras, 28
- gemm
  - ceras, 28, 29
- gemm\_cpu
  - ceras, 29
- get\_default\_session
  - ceras, 29
- glorot\_uniform
  - ceras, 29
- gradient
  - ceras::variable< T\_Operator >, 120
- gradient\_
  - ceras::variable\_state< T\_Operator >, 122
- gradient\_descent
  - ceras::gradient\_descent< Loss, T >, 77
- ground\_truth\_place\_holder\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, 74
- hadamard\_product
  - ceras, 29, 30
- hard\_sigmoid
  - ceras, 30
- has\_inf
  - ceras, 30
- has\_nan
  - ceras, 30
- Hinge
  - ceras, 53
- hinge\_loss
  - ceras, 30
- identity
  - ceras, 30
- img2col
  - ceras, 31
- includes.hpp
  - STB\_IMAGE\_IMPLEMENTATION, 134
  - STB\_IMAGE\_RESIZE\_IMPLEMENTATION, 135
  - STB\_IMAGE\_WRITE\_IMPLEMENTATION, 135
- Input
  - ceras, 31
- input
  - ceras::model< Ex, Ph >, 85
- input\_data\_
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 114
- input\_layer\_type
  - ceras::model< Ex, Ph >, 84
- input\_place\_holder\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, 74
- io\_layer\_type
  - ceras::compiled\_model< Model, Optimizer, Loss >, 71
- is\_binary\_operator\_v
  - ceras, 54
- is\_constant\_v
  - ceras, 54
- is\_place\_holder\_v
  - ceras, 54
- is\_tensor\_v
  - ceras, 54
- is\_unary\_operator\_v
  - ceras, 54
- is\_valid
  - ceras, 31
- is\_value\_v
  - ceras, 54
- is\_variable\_v
  - ceras, 55
- iterations\_
  - ceras::adadelta< Loss, T >, 62
  - ceras::adagrad< Loss, T >, 64
  - ceras::adam< Loss, T >, 67
  - ceras::rmsprop< Loss, T >, 94
  - ceras::sgd< Loss, T >, 101
- I1\_
  - ceras::regularizer< Float >, 92
- I2\_
  - ceras::regularizer< Float >, 92
- leaky\_relu
  - ceras, 31
- LeakyReLU
  - ceras, 31
- learning\_rate\_
  - ceras::adadelta< Loss, T >, 62
  - ceras::adagrad< Loss, T >, 64
  - ceras::adam< Loss, T >, 67
  - ceras::gradient\_descent< Loss, T >, 78
  - ceras::rmsprop< Loss, T >, 94
  - ceras::sgd< Loss, T >, 101
- lhs\_input\_data\_
  - ceras::binary\_operator< Lhs\_Operator, Rh\_Operator, Forward\_Action, Backward\_Action >, 69
- lhs\_op\_

- ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator>, 69
- Forward\_Action, Backward\_Action >, 69
- linspace
  - ceras, 31
- load\_data
  - ceras::dataset::fashion\_mnist, 59
  - ceras::dataset::mnist, 59
- load\_tensor
  - ceras, 32
- load\_weights
  - ceras::model< Ex, Ph >, 85
- log
  - ceras, 32
- loss\_
  - ceras::adadelta< Loss, T >, 62
  - ceras::adagrad< Loss, T >, 65
  - ceras::adam< Loss, T >, 67
  - ceras::compiled\_model< Model, Optimizer, Loss >, 74
  - ceras::gradient\_descent< Loss, T >, 78
  - ceras::rmsprop< Loss, T >, 94
  - ceras::sgd< Loss, T >, 101
- MAE
  - ceras, 55
- mae
  - ceras, 32
- make\_binary\_operator
  - ceras, 55
- make\_compiled\_model
  - ceras, 32
- make\_trainable
  - ceras, 32
- make\_unary\_operator
  - ceras, 55
- map
  - ceras::tensor< T, Allocator >, 107
- matrix
  - ceras, 18
- max
  - ceras, 33
- max\_pooling\_2d
  - ceras, 33
- maximum
  - ceras, 33
- MaxPooling2D
  - ceras, 33
- mean
  - ceras, 33, 34
- mean\_absolute\_error
  - ceras, 34
- mean\_reduce
  - ceras, 34
- mean\_squared\_error
  - ceras, 34
- mean\_squared\_logarithmic\_error
  - ceras, 35
- MeanAbsoluteError
  - ceras, 55
- MeanSquaredError
  - ceras, 56
- memory\_offset\_
  - ceras::tensor< T, Allocator >, 111
- min
  - ceras, 35
- minus
  - ceras, 35
- model
  - ceras::model< Ex, Ph >, 84
- model\_
  - ceras::compiled\_model< Model, Optimizer, Loss >, 74
- momentum\_
  - ceras::gradient\_descent< Loss, T >, 78
  - ceras::sgd< Loss, T >, 101
- MSE
  - ceras, 56
- mse
  - ceras, 35
- Multiply
  - ceras, 36
- multiply
  - ceras, 36
- ndim
  - ceras::tensor< T, Allocator >, 107
- negative
  - ceras, 36
- negative\_relu
  - ceras, 36
- nesterov\_
  - ceras::sgd< Loss, T >, 101
- norm
  - ceras, 37
- normalization\_batch
  - ceras, 37
- ones
  - ceras, 37
- ones\_like
  - ceras, 37
- op\_
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 114
- op\_type
  - ceras::tensor\_deduction< L, R >, 112
- Operator
  - ceras, 56
- operator!=
  - ceras, 38
- operator<
  - ceras, 40
- operator<<
  - ceras, 40
- operator<=
  - ceras, 40
- operator>
  - ceras, 41

operator>=  
     ceras, 41  
 operator\*  
     ceras, 38  
 operator\*=  
     ceras::tensor< T, Allocator >, 108  
 operator()  
     ceras::compiled\_model< Model, Optimizer, Loss >, 73  
     ceras::model< Ex, Ph >, 85  
 operator+  
     ceras, 38, 39  
 operator+=  
     ceras::tensor< T, Allocator >, 108  
 operator-  
     ceras, 39, 40  
     ceras::tensor< T, Allocator >, 108  
 operator-=  
     ceras::tensor< T, Allocator >, 108, 109  
 operator/  
     ceras, 40  
 operator/=  
     ceras::tensor< T, Allocator >, 109  
 operator=  
     ceras::ceras\_private::session< Tsor >, 97  
     ceras::place\_holder< Tsor >, 89, 90  
     ceras::tensor< T, Allocator >, 109  
     ceras::value< T >, 116, 117  
     ceras::variable< Tsor >, 120  
 operator==  
     ceras, 40, 41  
 operator[]  
     ceras::tensor< T, Allocator >, 109, 110  
     ceras::view\_2d< T >, 124  
     ceras::view\_3d< T >, 126, 127  
     ceras::view\_4d< T >, 129  
 optimizer\_  
     ceras::compiled\_model< Model, Optimizer, Loss >, 75  
 optimizer\_type  
     ceras::compiled\_model< Model, Optimizer, Loss >, 75  
 output  
     ceras::model< Ex, Ph >, 86  
 output\_data\_  
     ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 70  
     ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 114  
 output\_layer\_type  
     ceras::model< Ex, Ph >, 84  
 Place\_Holder  
     ceras, 57  
 place\_holder  
     ceras::place\_holder< Tsor >, 88, 89  
 place\_holder\_  
     ceras::model< Ex, Ph >, 87  
 place\_holder\_type  
     ceras::ceras\_private::session< Tsor >, 95  
 place\_holders\_  
     ceras::ceras\_private::session< Tsor >, 99  
 plus  
     ceras, 41  
 predict  
     ceras::compiled\_model< Model, Optimizer, Loss >, 73  
     ceras::model< Ex, Ph >, 86  
 randn  
     ceras, 41  
 randn\_like  
     ceras, 42  
 random  
     ceras, 42  
 random\_generator  
     ceras, 57  
 random\_like  
     ceras, 42  
 random\_normal\_like  
     ceras, 42  
 random\_seed  
     ceras, 57  
 read\_tensor  
     ceras, 43  
 rebind  
     ceras::ceras\_private::session< Tsor >, 97  
 reduce  
     ceras, 43  
 reduce\_mean  
     ceras, 43  
 reduce\_sum  
     ceras, 43, 44  
 regularizer  
     ceras::regularizer< Float >, 91  
 regularizer\_  
     ceras::variable< Tsor >, 121  
 ReLU  
     ceras, 44  
 relu  
     ceras, 44  
 remember  
     ceras::ceras\_private::session< Tsor >, 97  
 repeat  
     ceras, 44  
 replace\_placeholder\_with\_expression  
     ceras, 44  
 repmat  
     ceras, 45  
 reset  
     ceras::place\_holder< Tsor >, 90  
     ceras::tensor< T, Allocator >, 110  
     ceras::variable< Tsor >, 121  
 Reshape  
     ceras, 45  
 reshape  
     ceras, 45  
     ceras::tensor< T, Allocator >, 110

- resize
  - ceras::tensor< T, Allocator >, 110
- restore
  - ceras::ceras\_private::session< T, Allocator >, 98
- rho\_
  - ceras::adadelta< Loss, T >, 63
  - ceras::rmsprop< Loss, T >, 94
- rhs\_input\_data\_
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 70
- rhs\_op\_
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 70
- rms\_prop
  - ceras, 18
- RMSprop
  - ceras, 57
- rmsprop
  - ceras::rmsprop< Loss, T >, 93
- row\_
  - ceras::view\_2d< T >, 125
  - ceras::view\_3d< T >, 127
  - ceras::view\_4d< T >, 130
- run
  - ceras::ceras\_private::session< T, Allocator >, 98
- save
  - ceras::ceras\_private::session< T, Allocator >, 98
- save\_tensor
  - ceras, 45
- save\_weights
  - ceras::model< Ex, Ph >, 86
- self\_type
  - ceras::tensor< T, Allocator >, 103
- selu
  - ceras, 46
- serialize
  - ceras::ceras\_private::session< T, Allocator >, 98
- session
  - ceras::ceras\_private::session< T, Allocator >, 96
- SGD
  - ceras, 57
- sgd
  - ceras::sgd< Loss, T >, 100
- shape
  - ceras::constant< T, Allocator >, 76
  - ceras::tensor< T, Allocator >, 110
  - ceras::variable< T, Allocator >, 121
  - ceras::view\_2d< T >, 125
- shape\_
  - ceras::tensor< T, Allocator >, 111
- shape\_hint\_
  - ceras::place\_holder\_state< T, Allocator >, 90
- shared\_vector
  - ceras::tensor< T, Allocator >, 103
- shrink\_to
  - ceras::tensor< T, Allocator >, 111
- sigmoid
  - ceras, 46
- sign
  - ceras, 46
- size
  - ceras::tensor< T, Allocator >, 111
  - ceras::view\_2d< T >, 125
- slice
  - ceras::tensor< T, Allocator >, 111
- Softmax
  - ceras, 46
- softmax
  - ceras, 46, 47
- softplus
  - ceras, 47
- softsign
  - ceras, 47
- square
  - ceras, 47
- squared\_loss
  - ceras, 48
- squeeze
  - ceras, 48
- standard\_deviation
  - ceras, 48
- state\_
  - ceras::variable< T, Allocator >, 121
- STB\_IMAGE\_IMPLEMENTATION
  - includes.hpp, 134
- STB\_IMAGE\_RESIZE\_IMPLEMENTATION
  - includes.hpp, 135
- STB\_IMAGE\_WRITE\_IMPLEMENTATION
  - includes.hpp, 135
- std
  - ceras, 48
- Subtract
  - ceras, 48
- sum
  - ceras, 48, 49
- sum\_reduce
  - ceras, 49
- summary
  - ceras::model< Ex, Ph >, 87
- synchronized\_
  - ceras::regularizer< Float >, 92
- tanh
  - ceras, 49
- tap
  - ceras::ceras\_private::session< T, Allocator >, 98
- Tensor
  - ceras, 57
- tensor
  - ceras::tensor< T, Allocator >, 104, 105
- tensor\_type
  - ceras::adadelta< Loss, T >, 61
  - ceras::adagrad< Loss, T >, 63
  - ceras::adam< Loss, T >, 66
  - ceras::binary\_operator< Lhs\_Operator, Rhs\_Operator, Forward\_Action, Backward\_Action >, 68
  - ceras::gradient\_descent< Loss, T >, 77



- ceras::place\_holder< T, S, R >, 88
  - ceras::rmsprop< Loss, T >, 93
  - ceras::sgd< Loss, T >, 100
  - ceras::tensor\_deduction< L, R >, 112
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 114
  - ceras::variable< T, S, R >, 118
- tesseract
  - ceras, 18
- train\_on\_batch
  - ceras::compiled\_model< Model, Optimizer, Loss >, 73
- trainable
  - ceras::compiled\_model< Model, Optimizer, Loss >, 74
  - ceras::model< Ex, Ph >, 87
  - ceras::variable< T, S, R >, 121
- trainable\_
  - ceras::variable< T, S, R >, 122
- transpose
  - ceras, 49
- transposed\_
  - ceras::view\_2d< T >, 125
- truncated\_normal
  - ceras, 49
- Unary\_Operator
  - ceras, 58
- unary\_operator
  - ceras::unary\_operator< Operator, Forward\_Action, Backward\_Action >, 113
- up\_sampling\_2d
  - ceras, 49
- update\_cuda\_gemm\_threshold
  - ceras, 50
- UpSampling2D
  - ceras, 50
- Value
  - ceras, 58
- value
  - ceras::value< T >, 115, 116
- value\_type
  - ceras::regularizer< Float >, 91
  - ceras::tensor< T, Allocator >, 103
  - ceras::value< T >, 115
  - ceras::variable< T, S, R >, 118
- var
  - ceras, 50
- Variable
  - ceras, 58
- variable
  - ceras::variable< T, S, R >, 118, 119
- variable\_state\_type
  - ceras::ceras\_private::session< T, S, R >, 95
- variable\_type
  - ceras::ceras\_private::session< T, S, R >, 96
- variables\_
  - ceras::ceras\_private::session< T, S, R >, 99
- variance
  - ceras, 50
- vector\_
  - ceras::tensor< T, Allocator >, 111
- vector\_type
  - ceras::tensor< T, Allocator >, 104
- view\_2d
  - ceras::view\_2d< T >, 123, 124
- view\_3d
  - ceras::view\_3d< T >, 126
- view\_4d
  - ceras::view\_4d< T >, 128
- write\_tensor
  - ceras, 50
- zero\_padding\_2d
  - ceras, 50
- zeros
  - ceras, 51
- zeros\_like
  - ceras, 51