

ceras

Generated by Doxygen 1.9.1

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 ceras Namespace Reference	9
5.1.1 Typedef Documentation	18
5.1.1.1 ada_delta	18
5.1.1.2 ada_grad	18
5.1.1.3 cube	18
5.1.1.4 default_allocator	18
5.1.1.5 matrix	18
5.1.1.6 rms_prop	18
5.1.1.7 tesseract	19
5.1.2 Function Documentation	19
5.1.2.1 abs() [1/2]	19
5.1.2.2 abs() [2/2]	19
5.1.2.3 abs_loss()	19
5.1.2.4 Add()	19
5.1.2.5 add()	20
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()	21
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]	23
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()	24
5.1.2.25 conv2d()	24
5.1.2.26 copy()	25
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]	27
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()	29
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()	30
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()	31
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()	32
5.1.2.63 linspace()	32

5.1.2.64 load_tensor()	32
5.1.2.65 log()	32
5.1.2.66 mae()	32
5.1.2.67 make_compiled_model()	32
5.1.2.68 make_trainable()	33
5.1.2.69 max() [1/2]	33
5.1.2.70 max() [2/2]	33
5.1.2.71 max_pooling_2d()	33
5.1.2.72 maximum()	33
5.1.2.73 MaxPooling2D()	33
5.1.2.74 mean() [1/2]	34
5.1.2.75 mean() [2/2]	34
5.1.2.76 mean_absolute_error()	34
5.1.2.77 mean_reduce()	34
5.1.2.78 mean_squared_error()	34
5.1.2.79 mean_squared_logarithmic_error()	35
5.1.2.80 min() [1/2]	35
5.1.2.81 min() [2/2]	35
5.1.2.82 minus() [1/2]	35
5.1.2.83 minus() [2/2]	35
5.1.2.84 mse()	36
5.1.2.85 Multiply()	36
5.1.2.86 multiply() [1/2]	36
5.1.2.87 multiply() [2/2]	36
5.1.2.88 negative()	36
5.1.2.89 negative_relu()	37
5.1.2.90 norm()	37
5.1.2.91 normalization_batch()	37
5.1.2.92 ones()	37
5.1.2.93 ones_like() [1/2]	37
5.1.2.94 ones_like() [2/2]	38
5.1.2.95 operator!=(())	38
5.1.2.96 operator*() [1/4]	38
5.1.2.97 operator*() [2/4]	38
5.1.2.98 operator*() [3/4]	38
5.1.2.99 operator*() [4/4]	38
5.1.2.100 operator+() [1/4]	39
5.1.2.101 operator+() [2/4]	39
5.1.2.102 operator+() [3/4]	39
5.1.2.103 operator+() [4/4]	39
5.1.2.104 operator-() [1/4]	39
5.1.2.105 operator-() [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_max()	43
5.1.2.125 reduce_mean(1/2)	44
5.1.2.126 reduce_mean(2/2)	44
5.1.2.127 reduce_min()	44
5.1.2.128 reduce_sum(1/2)	44
5.1.2.129 reduce_sum(2/2)	45
5.1.2.130 relu()	45
5.1.2.131 ReLU()	45
5.1.2.132 repeat(1/2)	45
5.1.2.133 repeat(2/2)	45
5.1.2.134 replace_placeholder_with_expression()	46
5.1.2.135 repmat()	46
5.1.2.136 Reshape()	46
5.1.2.137 reshape(1/2)	46
5.1.2.138 reshape(2/2)	47
5.1.2.139 save_tensor()	47
5.1.2.140 selu()	47
5.1.2.141 sigmoid()	47
5.1.2.142 sign()	47
5.1.2.143 Softmax()	48
5.1.2.144 softmax(1/2)	48
5.1.2.145 softmax(2/2)	48
5.1.2.146 softplus()	48
5.1.2.147 softsign()	48

5.1.2.148 square()	48
5.1.2.149 squared_loss()	49
5.1.2.150 squeeze()	49
5.1.2.151 standard_deviation()	49
5.1.2.152 std()	49
5.1.2.153 Subtract()	50
5.1.2.154 sum() ^[1/2]	50
5.1.2.155 sum() ^[2/2]	50
5.1.2.156 sum_reduce()	50
5.1.2.157 tanh()	50
5.1.2.158 transpose()	51
5.1.2.159 truncated_normal()	51
5.1.2.160 up_sampling_2d()	51
5.1.2.161 update_cuda_gemm_threshold()	51
5.1.2.162 UpSampling2D()	51
5.1.2.163 var()	51
5.1.2.164 variance()	52
5.1.2.165 write_tensor()	52
5.1.2.166 zero_padding_2d()	52
5.1.2.167 zeros()	52
5.1.2.168 zeros_like()	53
5.1.3 Variable Documentation	53
5.1.3.1 Adadelta	53
5.1.3.2 Adagrad	53
5.1.3.3 Adam	53
5.1.3.4 Binary_Operator	54
5.1.3.5 BinaryCrossentropy	54
5.1.3.6 BinaryCrossEntropy	54
5.1.3.7 CategoricalCrossentropy	54
5.1.3.8 CategoricalCrossEntropy	55
5.1.3.9 Constant	55
5.1.3.10 Expression	55
5.1.3.11 Hinge	55
5.1.3.12 is_binary_operator_v	55
5.1.3.13 is_constant_v	56
5.1.3.14 is_place_holder_v	56
5.1.3.15 is_tensor_v	56
5.1.3.16 is_unary_operator_v	56
5.1.3.17 is_value_v	56
5.1.3.18 is_variable_v	56
5.1.3.19 MAE	57
5.1.3.20 make_binary_operator	57

5.1.3.21 make_unary_operator	57
5.1.3.22 MeanAbsoluteError	57
5.1.3.23 MeanSquaredError	58
5.1.3.24 MSE	58
5.1.3.25 Operator	58
5.1.3.26 Place_Holder	58
5.1.3.27 random_generator	59
5.1.3.28 random_seed	59
5.1.3.29 RMSprop	59
5.1.3.30 SGD	59
5.1.3.31 Tensor	59
5.1.3.32 Unary_Operator	60
5.1.3.33 Value	60
5.1.3.34 Variable	60
5.2 ceras::ceras_private Namespace Reference	60
5.3 ceras::dataset Namespace Reference	60
5.4 ceras::dataset::fashion_mnist Namespace Reference	60
5.4.1 Function Documentation	60
5.4.1.1 load_data()	60
5.5 ceras::dataset::mnist Namespace Reference	61
5.5.1 Function Documentation	61
5.5.1.1 load_data()	61
6 Class Documentation	63
6.1 ceras::adadelta< Loss, T > Struct Template Reference	63
6.1.1 Member Typedef Documentation	63
6.1.1.1 tensor_type	64
6.1.2 Constructor & Destructor Documentation	64
6.1.2.1 adadelta()	64
6.1.3 Member Function Documentation	64
6.1.3.1 forward()	64
6.1.4 Member Data Documentation	64
6.1.4.1 iterations_	64
6.1.4.2 learning_rate_	64
6.1.4.3 loss_	65
6.1.4.4 rho_	65
6.2 ceras::adagrad< Loss, T > Struct Template Reference	65
6.2.1 Member Typedef Documentation	65
6.2.1.1 tensor_type	66
6.2.2 Constructor & Destructor Documentation	66
6.2.2.1 adagrad()	66
6.2.3 Member Function Documentation	66

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

6.5.3 Member Function Documentation	74
6.5.3.1 evaluate()	74
6.5.3.2 fit()	74
6.5.3.3 operator()()	75
6.5.3.4 predict()	75
6.5.3.5 train_on_batch()	75
6.5.3.6 trainable()	76
6.5.4 Member Data Documentation	76
6.5.4.1 compiled_optimizer_	76
6.5.4.2 ground_truth_place_holder_	76
6.5.4.3 input_place_holder_	76
6.5.4.4 loss_	76
6.5.4.5 model_	77
6.5.4.6 optimizer_	77
6.5.4.7 optimizer_type	77
6.6 ceras::constant< Tsor > Struct Template Reference	77
6.6.1 Constructor & Destructor Documentation	78
6.6.1.1 constant()	78
6.6.2 Member Function Documentation	78
6.6.2.1 backward()	78
6.6.2.2 forward()	78
6.6.2.3 shape()	78
6.6.3 Member Data Documentation	78
6.6.3.1 data_	78
6.7 ceras::gradient_descent< Loss, T > Struct Template Reference	79
6.7.1 Member Typedef Documentation	79
6.7.1.1 tensor_type	79
6.7.2 Constructor & Destructor Documentation	79
6.7.2.1 gradient_descent()	79
6.7.3 Member Function Documentation	80
6.7.3.1 forward()	80
6.7.4 Member Data Documentation	80
6.7.4.1 learning_rate_	80
6.7.4.2 loss_	80
6.7.4.3 momentum_	80
6.8 ceras::is_binary_operator< T > Struct Template Reference	80
6.9 ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > > Struct Template Reference	81
6.10 ceras::is_constant< T > Struct Template Reference	81
6.11 ceras::is_constant< constant< Tsor > > Struct Template Reference	81
6.12 ceras::is_place_holder< T > Struct Template Reference	82
6.13 ceras::is_place_holder< place_holder< Tsor > > Struct Template Reference	82

6.14	ceras::is_tensor< T > Struct Template Reference	82
6.15	ceras::is_tensor< tensor< T, A > > Struct Template Reference	83
6.16	ceras::is_unary_operator< T > Struct Template Reference	83
6.17	ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > > Struct Template Reference	83
6.18	ceras::is_value< T > Struct Template Reference	84
6.19	ceras::is_value< value< T > > Struct Template Reference	84
6.20	ceras::is_variable< T > Struct Template Reference	84
6.21	ceras::is_variable< variable< Tsor > > Struct Template Reference	85
6.22	ceras::model< Ex, Ph > Struct Template Reference	85
6.22.1	Detailed Description	86
6.22.2	Member Typedef Documentation	86
6.22.2.1	input_layer_type	86
6.22.2.2	output_layer_type	86
6.22.3	Constructor & Destructor Documentation	86
6.22.3.1	model() [1/2]	86
6.22.3.2	model() [2/2]	87
6.22.4	Member Function Documentation	87
6.22.4.1	compile()	87
6.22.4.2	input() [1/2]	88
6.22.4.3	input() [2/2]	88
6.22.4.4	load_weights()	88
6.22.4.5	operator>()	88
6.22.4.6	output() [1/2]	89
6.22.4.7	output() [2/2]	89
6.22.4.8	predict() [1/3]	89
6.22.4.9	predict() [2/3]	89
6.22.4.10	predict() [3/3]	90
6.22.4.11	save_weights()	90
6.22.4.12	summary()	90
6.22.4.13	trainable()	91
6.22.5	Member Data Documentation	91
6.22.5.1	expression_	91
6.22.5.2	input_layer_	91
6.22.5.3	output_layer_	91
6.22.5.4	place_holder_	91
6.23	ceras::place_holder< Tsor > Struct Template Reference	92
6.23.1	Member Typedef Documentation	92
6.23.1.1	tensor_type	92
6.23.2	Constructor & Destructor Documentation	92
6.23.2.1	place_holder() [1/4]	92
6.23.2.2	place_holder() [2/4]	93

6.23.2.3 place_holder() [3 / 4]	93
6.23.2.4 place_holder() [4 / 4]	93
6.23.3 Member Function Documentation	93
6.23.3.1 backward()	93
6.23.3.2 bind()	93
6.23.3.3 forward()	93
6.23.3.4 operator=() [1 / 2]	94
6.23.3.5 operator=() [2 / 2]	94
6.23.3.6 reset()	94
6.24 ceras::place_holder_state< T> Struct Template Reference	94
6.24.1 Member Data Documentation	94
6.24.1.1 data_	94
6.24.1.2 shape_hint_	95
6.25 ceras::regularizer< Float> Struct Template Reference	95
6.25.1 Member Typedef Documentation	95
6.25.1.1 value_type	95
6.25.2 Constructor & Destructor Documentation	95
6.25.2.1 regularizer()	96
6.25.3 Member Data Documentation	96
6.25.3.1 l1_	96
6.25.3.2 l2_	96
6.25.3.3 synchronized_	96
6.26 ceras::rmsprop< Loss, T> Struct Template Reference	96
6.26.1 Member Typedef Documentation	97
6.26.1.1 tensor_type	97
6.26.2 Constructor & Destructor Documentation	97
6.26.2.1 rmsprop()	97
6.26.3 Member Function Documentation	97
6.26.3.1 forward()	98
6.26.4 Member Data Documentation	98
6.26.4.1 decay_	98
6.26.4.2 iterations_	98
6.26.4.3 learning_rate_	98
6.26.4.4 loss_	98
6.26.4.5 rho_	98
6.27 ceras::ceras_private::session< T> Struct Template Reference	99
6.27.1 Member Typedef Documentation	99
6.27.1.1 place_holder_type	99
6.27.1.2 variable_state_type	100
6.27.1.3 variable_type	100
6.27.2 Constructor & Destructor Documentation	100
6.27.2.1 session() [1 / 3]	100

6.27.2.2 session() [2/3]	100
6.27.2.3 session() [3/3]	100
6.27.2.4 ~session()	100
6.27.3 Member Function Documentation	101
6.27.3.1 bind()	101
6.27.3.2 deserialize()	101
6.27.3.3 operator=() [1/2]	101
6.27.3.4 operator=() [2/2]	101
6.27.3.5 rebind()	101
6.27.3.6 remember()	102
6.27.3.7 restore()	102
6.27.3.8 run()	102
6.27.3.9 save()	102
6.27.3.10 serialize()	102
6.27.3.11 tap()	102
6.27.4 Member Data Documentation	103
6.27.4.1 place_holders_	103
6.27.4.2 variables_	103
6.28 ceras::sgd< Loss, T > Struct Template Reference	103
6.28.1 Member Typedef Documentation	104
6.28.1.1 tensor_type	104
6.28.2 Constructor & Destructor Documentation	104
6.28.2.1 sgd()	104
6.28.3 Member Function Documentation	104
6.28.3.1 forward()	104
6.28.4 Member Data Documentation	104
6.28.4.1 decay_	105
6.28.4.2 iterations_	105
6.28.4.3 learning_rate_	105
6.28.4.4 loss_	105
6.28.4.5 momentum_	105
6.28.4.6 nesterov_	105
6.29 ceras::tensor< T, Allocator > Struct Template Reference	106
6.29.1 Member Typedef Documentation	107
6.29.1.1 allocator	107
6.29.1.2 self_type	107
6.29.1.3 shared_vector	107
6.29.1.4 value_type	108
6.29.1.5 vector_type	108
6.29.2 Constructor & Destructor Documentation	108
6.29.2.1 tensor() [1/7]	108
6.29.2.2 tensor() [2/7]	108

6.29.2.3 <code>tensor()</code> [3/7]	108
6.29.2.4 <code>tensor()</code> [4/7]	108
6.29.2.5 <code>tensor()</code> [5/7]	109
6.29.2.6 <code>tensor()</code> [6/7]	109
6.29.2.7 <code>tensor()</code> [7/7]	109
6.29.3 Member Function Documentation	109
6.29.3.1 <code>as_scalar()</code>	109
6.29.3.2 <code>as_type()</code>	109
6.29.3.3 <code>begin()</code> [1/2]	109
6.29.3.4 <code>begin()</code> [2/2]	110
6.29.3.5 <code>cbegin()</code>	110
6.29.3.6 <code>cend()</code>	110
6.29.3.7 <code>copy()</code>	110
6.29.3.8 <code>creep_to()</code>	110
6.29.3.9 <code>data()</code> [1/2]	110
6.29.3.10 <code>data()</code> [2/2]	110
6.29.3.11 <code>deep_copy()</code> [1/2]	111
6.29.3.12 <code>deep_copy()</code> [2/2]	111
6.29.3.13 <code>empty()</code>	111
6.29.3.14 <code>end()</code> [1/2]	111
6.29.3.15 <code>end()</code> [2/2]	111
6.29.3.16 <code>map()</code>	111
6.29.3.17 <code>ndim()</code>	112
6.29.3.18 <code>operator*=()</code> [1/2]	112
6.29.3.19 <code>operator*=()</code> [2/2]	112
6.29.3.20 <code>operator+=()</code> [1/2]	112
6.29.3.21 <code>operator+=()</code> [2/2]	112
6.29.3.22 <code>operator-()</code>	112
6.29.3.23 <code>operator-=()</code> [1/2]	113
6.29.3.24 <code>operator-=()</code> [2/2]	113
6.29.3.25 <code>operator/=()</code> [1/2]	113
6.29.3.26 <code>operator/=()</code> [2/2]	113
6.29.3.27 <code>operator=()</code> [1/2]	113
6.29.3.28 <code>operator=()</code> [2/2]	113
6.29.3.29 <code>operator[]()</code> [1/2]	114
6.29.3.30 <code>operator[]()</code> [2/2]	114
6.29.3.31 <code>reset()</code>	114
6.29.3.32 <code>reshape()</code>	114
6.29.3.33 <code>resize()</code>	114
6.29.3.34 <code>shape()</code>	115
6.29.3.35 <code>shrink_to()</code>	115
6.29.3.36 <code>size()</code>	115

6.29.3.37 slice()	115
6.29.4 Member Data Documentation	115
6.29.4.1 memory_offset_	115
6.29.4.2 shape_	115
6.29.4.3 vector_	116
6.30 ceras::tensor_deduction< L, R > Struct Template Reference	116
6.30.1 Member Typedef Documentation	116
6.30.1.1 op_type	116
6.30.1.2 tensor_type	116
6.31 ceras::unary_operator< Operator, Forward_Action, Backward_Action > Struct Template Reference	116
6.31.1 Constructor & Destructor Documentation	117
6.31.1.1 unary_operator()	117
6.31.2 Member Function Documentation	117
6.31.2.1 backward()	117
6.31.2.2 forward()	117
6.31.3 Member Data Documentation	118
6.31.3.1 backward_action_	118
6.31.3.2 forward_action_	118
6.31.3.3 input_data_	118
6.31.3.4 op_	118
6.31.3.5 output_data_	118
6.31.3.6 tensor_type	118
6.32 ceras::value< T > Struct Template Reference	119
6.32.1 Member Typedef Documentation	119
6.32.1.1 value_type	119
6.32.2 Constructor & Destructor Documentation	119
6.32.2.1 value() [1/4]	120
6.32.2.2 value() [2/4]	120
6.32.2.3 value() [3/4]	120
6.32.2.4 value() [4/4]	120
6.32.3 Member Function Documentation	120
6.32.3.1 backward()	120
6.32.3.2 forward()	120
6.32.3.3 operator=() [1/2]	121
6.32.3.4 operator=() [2/2]	121
6.32.4 Member Data Documentation	121
6.32.4.1 data_	121
6.33 ceras::variable< Tsr > Struct Template Reference	121
6.33.1 Member Typedef Documentation	122
6.33.1.1 tensor_type	122
6.33.1.2 value_type	122
6.33.2 Constructor & Destructor Documentation	122

6.33.2.1 variable() [1/4]	123
6.33.2.2 variable() [2/4]	123
6.33.2.3 variable() [3/4]	123
6.33.2.4 variable() [4/4]	123
6.33.3 Member Function Documentation	123
6.33.3.1 backward()	123
6.33.3.2 contexts() [1/2]	123
6.33.3.3 contexts() [2/2]	124
6.33.3.4 data() [1/2]	124
6.33.3.5 data() [2/2]	124
6.33.3.6 forward()	124
6.33.3.7 gradient() [1/2]	124
6.33.3.8 gradient() [2/2]	124
6.33.3.9 operator=() [1/2]	124
6.33.3.10 operator=() [2/2]	125
6.33.3.11 reset()	125
6.33.3.12 shape()	125
6.33.3.13 trainable() [1/2]	125
6.33.3.14 trainable() [2/2]	125
6.33.4 Member Data Documentation	125
6.33.4.1 regularizer_	125
6.33.4.2 state_	126
6.33.4.3 trainable_	126
6.34 ceras::variable_state< Tsor > Struct Template Reference	126
6.34.1 Member Data Documentation	126
6.34.1.1 contexts_	126
6.34.1.2 data_	126
6.34.1.3 gradient_	127
6.35 ceras::view_2d< T > Struct Template Reference	127
6.35.1 Member Typedef Documentation	128
6.35.1.1 col_type	128
6.35.1.2 const_col_type	128
6.35.1.3 const_row_type	128
6.35.1.4 row_type	128
6.35.1.5 value_type	128
6.35.2 Constructor & Destructor Documentation	128
6.35.2.1 view_2d() [1/3]	129
6.35.2.2 view_2d() [2/3]	129
6.35.2.3 view_2d() [3/3]	129
6.35.3 Member Function Documentation	129
6.35.3.1 begin()	129
6.35.3.2 col()	129

6.35.3.3 col_begin() [1/2]	130
6.35.3.4 col_begin() [2/2]	130
6.35.3.5 col_end() [1/2]	130
6.35.3.6 col_end() [2/2]	130
6.35.3.7 data() [1/2]	130
6.35.3.8 data() [2/2]	130
6.35.3.9 end()	131
6.35.3.10 operator[]() [1/2]	131
6.35.3.11 operator[]() [2/2]	131
6.35.3.12 row()	131
6.35.3.13 row_begin() [1/2]	131
6.35.3.14 row_begin() [2/2]	131
6.35.3.15 row_end() [1/2]	132
6.35.3.16 row_end() [2/2]	132
6.35.3.17 shape()	132
6.35.3.18 size()	132
6.35.4 Member Data Documentation	132
6.35.4.1 col_	132
6.35.4.2 data_	132
6.35.4.3 row_	133
6.35.4.4 transposed_	133
6.36 ceras::view_3d< T > Struct Template Reference	133
6.36.1 Constructor & Destructor Documentation	133
6.36.1.1 view_3d()	133
6.36.2 Member Function Documentation	134
6.36.2.1 operator[]() [1/2]	134
6.36.2.2 operator[]() [2/2]	134
6.36.3 Member Data Documentation	134
6.36.3.1 channel_	134
6.36.3.2 col_	134
6.36.3.3 data_	134
6.36.3.4 row_	135
6.37 ceras::view_4d< T > Struct Template Reference	135
6.37.1 Detailed Description	135
6.37.2 Constructor & Destructor Documentation	135
6.37.2.1 view_4d()	136
6.37.3 Member Function Documentation	137
6.37.3.1 operator[]() [1/2]	137
6.37.3.2 operator[]() [2/2]	137
6.37.4 Member Data Documentation	138
6.37.4.1 batch_size_	138
6.37.4.2 channel_	138

6.37.4.3 col_	138
6.37.4.4 data_	138
6.37.4.5 row_	138
7 File Documentation	139
7.1 /data/structured_folders/workspace/github.repo/ceras/include/activation.hpp File Reference	139
7.2 /data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp File Reference	140
7.3 /data/structured_folders/workspace/github.repo/ceras/include/config.hpp File Reference	140
7.4 /data/structured_folders/workspace/github.repo/ceras/include/constant.hpp File Reference	140
7.5 /data/structured_folders/workspace/github.repo/ceras/include/dataset.hpp File Reference	141
7.6 /data/structured_folders/workspace/github.repo/ceras/include/includes.hpp File Reference	142
7.6.1 Macro Definition Documentation	142
7.6.1.1 STB_IMAGE_IMPLEMENTATION	143
7.6.1.2 STB_IMAGE_RESIZE_IMPLEMENTATION	143
7.6.1.3 STB_IMAGE_WRITE_IMPLEMENTATION	143
7.7 /data/structured_folders/workspace/github.repo/ceras/include/layer.hpp File Reference	143
7.8 /data/structured_folders/workspace/github.repo/ceras/include/loss.hpp File Reference	144
7.9 /data/structured_folders/workspace/github.repo/ceras/include/model.hpp File Reference	145
7.10 /data/structured_folders/workspace/github.repo/ceras/include/operation.hpp File Reference	146
7.11 /data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp File Reference	149
7.12 /data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp File Reference	150
7.13 /data/structured_folders/workspace/github.repo/ceras/include/session.hpp File Reference	151
7.14 /data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp File Reference	151
7.15 /data/structured_folders/workspace/github.repo/ceras/include/value.hpp File Reference	155
7.16 /data/structured_folders/workspace/github.repo/ceras/include/variable.hpp File Reference	156
7.17 /data/structured_folders/workspace/github.repo/ceras/include/xmodel.hpp File Reference	157
Index	159

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

ceras	9
ceras::ceras_private	60
ceras::dataset	60
ceras::dataset::fashion_mnist	60
ceras::dataset::mnist	61

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

ceras::compiled_model< Model, Optimizer, Loss >	72
enable_id	
ceras::adadelta< Loss, T >	63
ceras::adagrad< Loss, T >	65
ceras::adam< Loss, T >	67
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >	69
ceras::constant< Tsor >	77
ceras::gradient_descent< Loss, T >	79
ceras::place_holder< Tsor >	92
ceras::rmsprop< Loss, T >	96
ceras::sgd< Loss, T >	103
ceras::tensor< T, Allocator >	106
ceras::unary_operator< Operator, Forward_Action, Backward_Action >	116
ceras::value< T >	119
ceras::variable< Tsor >	121
enable_shared	
ceras::adadelta< Loss, T >	63
ceras::adagrad< Loss, T >	65
ceras::adam< Loss, T >	67
ceras::gradient_descent< Loss, T >	79
ceras::rmsprop< Loss, T >	96
ceras::sgd< Loss, T >	103
enable_shared_state	
ceras::place_holder< Tsor >	92
std::false_type	
ceras::is_binary_operator< T >	80
ceras::is_constant< T >	81
ceras::is_place_holder< T >	82
ceras::is_tensor< T >	82
ceras::is_unary_operator< T >	83
ceras::is_value< T >	84
ceras::is_variable< T >	84
ceras::model< Ex, Ph >	85
ceras::place_holder_state< Tsor >	94
ceras::regularizer< Float >	95

ceras::regularizer< value_type >	95
ceras::ceras_private::session< Tsor >	99
ceras::tensor_deduction< L, R >	116
ceras::tensor_deduction< Lhs_Operator, Rhs_Operator >	116
std::true_type	
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > >	81
ceras::is_constant< constant< Tsor > >	81
ceras::is_place_holder< place_holder< Tsor > >	82
ceras::is_tensor< tensor< T, A > >	83
ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >	83
ceras::is_value< value< T > >	84
ceras::is_variable< variable< Tsor > >	85
ceras::variable_state< Tsor >	126
ceras::view_2d< T >	127
ceras::view_3d< T >	133
ceras::view_4d< T >	135

Chapter 3

Class Index

3.1 Class List

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

ceras::adadelta< Loss, T >	63
ceras::adagrad< Loss, T >	65
ceras::adam< Loss, T >	67
ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >	69
ceras::compiled_model< Model, Optimizer, Loss >	72
ceras::constant< Tsor >	77
ceras::gradient_descent< Loss, T >	79
ceras::is_binary_operator< T >	80
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > >	81
ceras::is_constant< T >	81
ceras::is_constant< constant< Tsor > >	81
ceras::is_place_holder< T >	82
ceras::is_place_holder< place_holder< Tsor > >	82
ceras::is_tensor< T >	82
ceras::is_tensor< tensor< T, A > >	83
ceras::is_unary_operator< T >	83
ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >	83
ceras::is_value< T >	84
ceras::is_value< value< T > >	84
ceras::is_variable< T >	84
ceras::is_variable< variable< Tsor > >	85
ceras::model< Ex, Ph >	85
ceras::place_holder< Tsor >	92
ceras::place_holder_state< Tsor >	94
ceras::regularizer< Float >	95
ceras::rmsprop< Loss, T >	96
ceras::ceras_private::session< Tsor >	99
ceras::sgd< Loss, T >	103
ceras::tensor< T, Allocator >	106
ceras::tensor_deduction< L, R >	116
ceras::unary_operator< Operator, Forward_Action, Backward_Action >	116
ceras::value< T >	119
ceras::variable< Tsor >	121
ceras::variable_state< Tsor >	126
ceras::view_2d< T >	127
ceras::view_3d< T >	133
ceras::view_4d< T >	135

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	139
/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp	140
/data/structured_folders/workspace/github.repo/ceras/include/config.hpp	140
/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp	140
/data/structured_folders/workspace/github.repo/ceras/include/dataset.hpp	141
/data/structured_folders/workspace/github.repo/ceras/include/includes.hpp	142
/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp	143
/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp	144
/data/structured_folders/workspace/github.repo/ceras/include/model.hpp	145
/data/structured_folders/workspace/github.repo/ceras/include/operation.hpp	146
/data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp	149
/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp	150
/data/structured_folders/workspace/github.repo/ceras/include/session.hpp	151
/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp	151
/data/structured_folders/workspace/github.repo/ceras/include/value.hpp	155
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp	156
/data/structured_folders/workspace/github.repo/ceras/include/xmodel.hpp	157

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 Rhs_Expression>
constexpr auto [mean_squared_logarithmic_error](#) (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto [squared_loss](#) (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto [mean_squared_error](#) (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto [mse](#) (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto [abs_loss](#) (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto [mean_absolute_error](#) (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto [mae](#) (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept

- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto cross_entropy (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto binary_cross_entropy_loss (Lhs_Expression const &ground_truth, Rhs_Expression const &prediction) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto cross_entropy_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto hinge_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Ex>`
`void make_trainable (Ex &ex, bool t)`
- `template<Expression Ex, Place_Holder Ph, Expression Ey>`
`auto replace_placeholder_with_expression (Ex const &ex, Ph const &old_place_holder, Ey const &new_↵
expression)`
- `template<typename Model , typename Optimizer , typename Loss >`
`auto make_compiled_model (Model const &m, Loss const &l, Optimizer const &o)`
- `template<Expression Ex>`
`std::string computation_graph (Ex const &ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto plus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto operator+ (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression 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=1`, `unsigned long const col_stride=1`, `unsigned long const row_dilation=1`, `unsigned long const col_dilation=1`) noexcept
- auto `conv2d` (`unsigned long row_input`, `unsigned long col_input`, `unsigned long const row_stride=1`, `unsigned long const col_stride=1`, `unsigned long const row_dilation=1`, `unsigned long const col_dilation=1`, `std::string const &padding="valid"`) noexcept
- `template<typename T >`
requires `std::floating_point< T >` auto `drop_out` (`T const factor`) noexcept
- auto `max_pooling_2d` (`unsigned long stride`) noexcept
- auto `average_pooling_2d` (`unsigned long stride`) noexcept
- auto `up_sampling_2d` (`unsigned long stride`) noexcept
- `template<typename T = double>`
requires `std::floating_point< T >` auto `normalization_batch` (`T const momentum=0.98`) noexcept
- `template<typename T >`
requires `std::floating_point< T >` auto `batch_normalization` (`T const momentum=0.98`) noexcept
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
constexpr auto `concatenate` (`Lhs_Expression const &lhs_ex`, `Rhs_Expression const &rhs_ex`) noexcept
- auto `concatenate` (`unsigned long axis=-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 axis=-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).
- auto `repeat` (`unsigned long repeats`, `unsigned long axis=-1`) noexcept
Repeats elements along an axis.
- auto `reduce_min` (`unsigned long axis=-1`) noexcept
Reduce minimal elements along an axis.
- auto `reduce_max` (`unsigned long axis=-1`) noexcept
Reduce maximum elements along an axis.
- `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

<i>ex</i>	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 Rhs_Expression>
constexpr auto ceras::concat (
    Lhs_Expression const & lhs_ex,
    Rhs_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 Rhs_Expression>
constexpr auto ceras::mae (
    Lhs_Expression const & lhs_ex,
    Rhs_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_max()

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

Reduce maximum elements along an axis.

Parameters

<i>axis</i>	The axis along which to reduce maximum values. Defaults to the last axis.
-------------	---

Example code:

```

auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = reduce_max( 0 )( a ); // <- output shape is ( 3, 5 )
auto b = reduce_max( 1 )( a ); // <- output shape is ( 2, 5 )
auto b = reduce_max( 2 )( a ); // <- output shape is ( 2, 3 )
auto b = reduce_max( ) ( a ); // <- output shape is ( 2, 3 )

```

5.1.2.125 reduce_mean() [1/2]

```

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

```

5.1.2.126 reduce_mean() [2/2]

```

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

```

5.1.2.127 reduce_min()

```

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

```

Reduce minimal elements along an axis.

Parameters

<i>axis</i>	The axis along which to reduce minimal values. Defaults to the last axis.
-------------	---

Example code:

```

auto a = variable{ random<float>( {2, 3, 5} ) };
auto b = reduce_min( 0 )( a ); // <- output shape is ( 3, 5 )
auto b = reduce_min( 1 )( a ); // <- output shape is ( 2, 5 )
auto b = reduce_min( 2 )( a ); // <- output shape is ( 2, 3 )
auto b = reduce_min( ) ( a ); // <- output shape is ( 2, 3 )

```

5.1.2.128 reduce_sum() [1/2]

```

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

```

5.1.2.129 reduce_sum() [2/2]

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

5.1.2.130 relu()

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

5.1.2.131 ReLU()

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

Rectified Linear Unit activation function.

5.1.2.132 repeat() [1/2]

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

5.1.2.133 repeat() [2/2]

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

Repeats elements along an axis.

Parameters

<i>repeats</i>	The number of repetitions for each element.
<i>axis</i>	The axis along which to repeat values. Defaults to the last axis.

Example code:

```
auto a = variable{ random<float>( {2, 3, 5} )};
```

```

auto b0 = repeat( 2, 0 )( a ); // <- output shape is ( 4, 3, 5 )
auto b1 = repeat( 2, 1 )( a ); // <- output shape is ( 2, 6, 5 )
auto b2 = repeat( 2, 2 )( a ); // <- output shape is ( 2, 3, 10 )
auto bx = repeat( 2 )( a ); // <- output shape is ( 2, 3, 10 )

```

5.1.2.134 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, place_holder , a constant or a value
<i>old_place_holder</i>	An place holder in <i>ex</i>
<i>new_expression</i>	An expression that will replace <i>old_place_holder</i> in <i>ex</i> .

Returns

A expression inheriting the topology of *ex*, but with *old_place_holder* replaced by *new_expression*

5.1.2.135 repmat()

```

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

```

5.1.2.136 Reshape()

```

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

```

Reshapes inputs into the given shape.

5.1.2.137 reshape() [1/2]

```

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

```


5.1.2.138 reshape() [2/2]

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

5.1.2.139 save_tensor()

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

5.1.2.140 selu()

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

5.1.2.141 sigmoid()

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

5.1.2.142 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);
```

5.1.2.143 Softmax()

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

Softmax activation function.

5.1.2.144 softmax() [1/2]

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

5.1.2.145 softmax() [2/2]

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

5.1.2.146 softplus()

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

5.1.2.147 softsign()

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

5.1.2.148 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);
```

5.1.2.149 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]
```

5.1.2.150 squeeze()

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

5.1.2.151 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]
```

5.1.2.152 std()

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

5.1.2.153 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 };
```

5.1.2.154 sum() [1/2]

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

5.1.2.155 sum() [2/2]

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

5.1.2.156 sum_reduce()

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

5.1.2.157 tanh()

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

5.1.2.158 transpose()

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

5.1.2.159 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} )
```

5.1.2.160 up_sampling_2d()

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

5.1.2.161 update_cuda_gemm_threshold()

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

5.1.2.162 UpSampling2D()

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

Upsampling layer for 2D inputs.

5.1.2.163 var()

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

5.1.2.164 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]
```

5.1.2.165 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 )
```

5.1.2.166 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)
```

5.1.2.167 zeros()

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

5.1.2.168 zeros_like()

```
template<Tensor Tsr>
constexpr Tsr ceras::zeros_like (
    Tsr const & tsr ) [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.

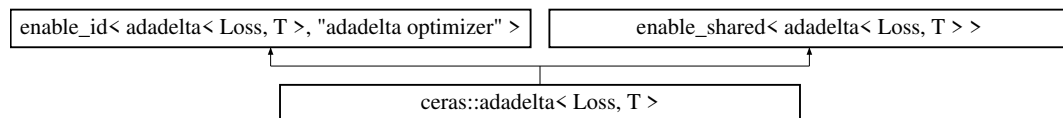
Chapter 6

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_
```

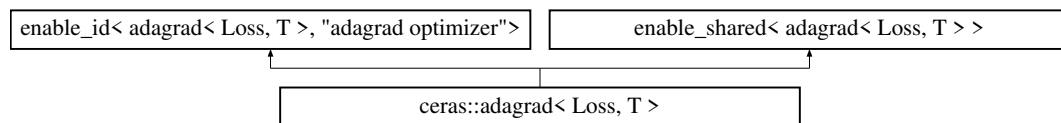
The documentation for this struct was generated from the following file:

- /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_
```

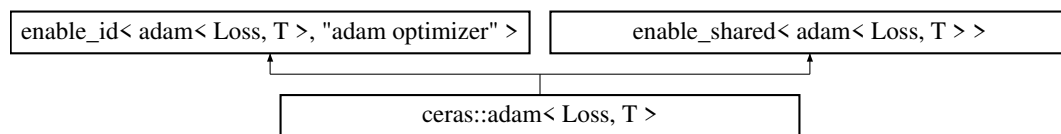
The documentation for this struct was generated from the following file:

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

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_
```

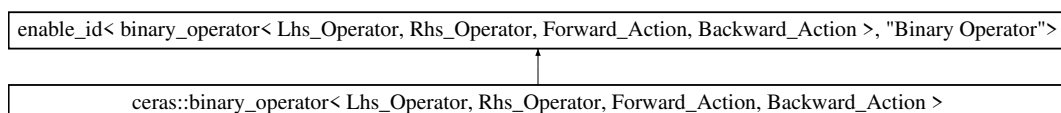
The documentation for this struct was generated from the following file:

- /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
```

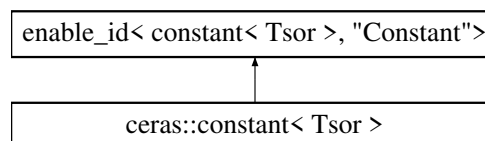
The documentation for this struct was generated from the following file:

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

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_
```

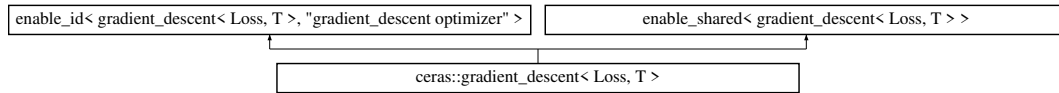
The documentation for this struct was generated from the following file:

- /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_
```

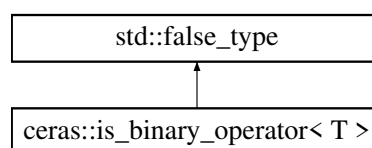
The documentation for this struct was generated from the following file:

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

6.8 ceras::is_binary_operator< T > Struct Template Reference

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

Inheritance diagram for ceras::is_binary_operator< T >:



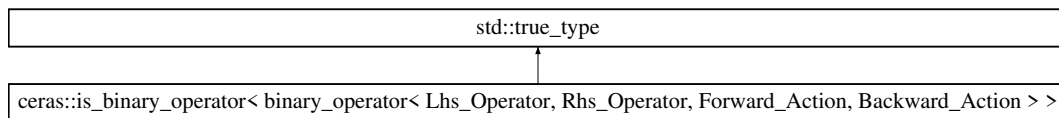
The documentation for this struct was generated from the following file:

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

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 > >:



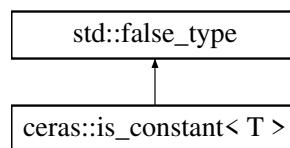
The documentation for this struct was generated from the following file:

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

6.10 ceras::is_constant< T > Struct Template Reference

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

Inheritance diagram for ceras::is_constant< T >:



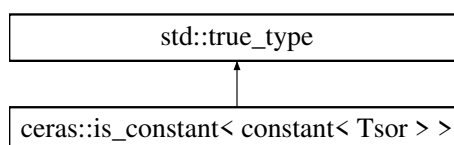
The documentation for this struct was generated from the following file:

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

6.11 ceras::is_constant< constant< Tsor > > Struct Template Reference

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

Inheritance diagram for ceras::is_constant< constant< Tsor > >:



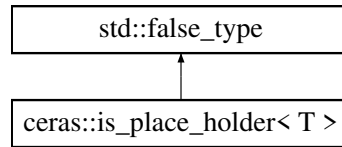
The documentation for this struct was generated from the following file:

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

6.12 ceras::is_place_holder< T > Struct Template Reference

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

Inheritance diagram for ceras::is_place_holder< T >:



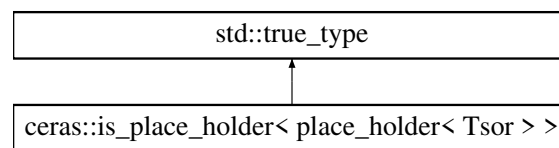
The documentation for this struct was generated from the following file:

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

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 > >:



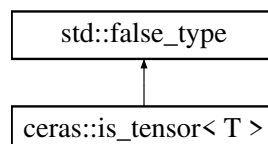
The documentation for this struct was generated from the following file:

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

6.14 ceras::is_tensor< T > Struct Template Reference

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

Inheritance diagram for ceras::is_tensor< T >:



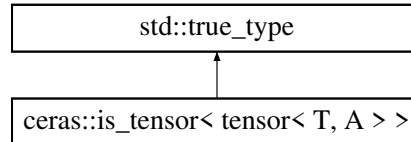
The documentation for this struct was generated from the following file:

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

6.15 ceras::is_tensor< tensor< T, A > > Struct Template Reference

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

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



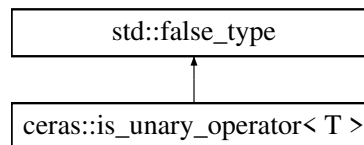
The documentation for this struct was generated from the following file:

- /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 >:



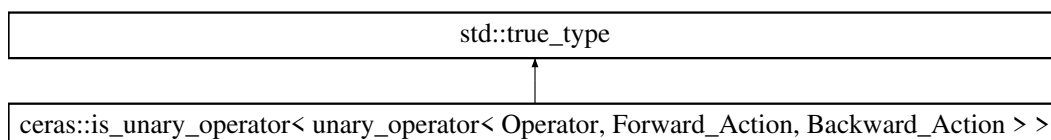
The documentation for this struct was generated from the following file:

- /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 > >:



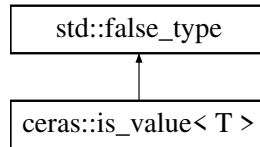
The documentation for this struct was generated from the following file:

- /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 >:



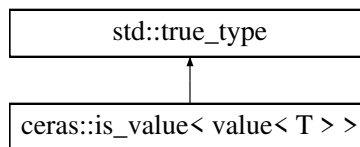
The documentation for this struct was generated from the following file:

- /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 > >:



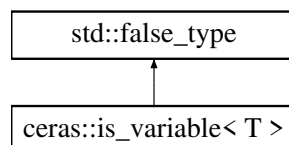
The documentation for this struct was generated from the following file:

- /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 >:



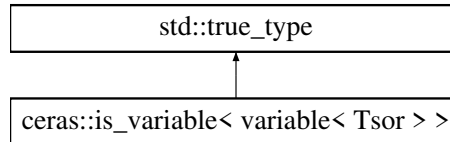
The documentation for this struct was generated from the following file:

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

6.21 ceras::is_variable< variable< T_{sr} > > Struct Template Reference

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

Inheritance diagram for ceras::is_variable< variable< T_{sr} > >:



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 T_{sr}>
auto [predict](#) (T_{sr} const &input_tensor)
- template<Expression Exp>
auto [operator](#)() (Exp const &ex) const noexcept
- template<typename Loss , typename Optimizer >
auto [compile](#) (Loss const &l, Optimizer const &o)
- void [trainable](#) (bool t)
- void [save_weights](#) (std::string const &file)
- void [load_weights](#) (std::string const &file)
- void [summary](#) (std::string const &file_name=std::string{}) const noexcept
- constexpr [model](#) (Input_List const &input_layer, Output_List const &output_layer)
- constexpr auto [input](#) () const
Returns the input layer(s) of the model in a 'list', which is are [place_holders](#).
- constexpr auto [output](#) () const
Returns the output layer(s) of the model in a 'list', which is are expressions.
- template<Tensor T_{sr}>
constexpr auto [predict](#) (T_{sr} const &input_tensor) const
- template<List T_{sr}_List>
auto [predict](#) (T_{sr}_List const &input_tensor) const

Public Attributes

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

6.22.1 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() [1/2]

```
template<Expression Ex, Place_Holder Ph>
ceras::model< Ex, Ph >::model (
    input\_layer\_type const & place_holder,
    output\_layer\_type const & expression ) [inline]
```


Parameters

<i>place_holder</i>	The input layer of the model, a place holder.
<i>expression</i>	The output layer of the model, a expression template.

Example code to generate a model:

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

6.22.3.2 model() [2/2]

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

6.22.4 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 useage:

```
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() [1/2]

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

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

6.22.4.3 input() [2/2]

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

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

6.22.4.4 load_weights()

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

Loads all variables from a file

6.22.4.5 operator>()

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

Generating a new expression by using the current model.

Parameters

ex	An expression that represents the input to the model.
----	---

Returns

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

Example code

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

6.22.4.6 output() [1/2]

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

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

6.22.4.7 output() [2/2]

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

Returns the output layer of the model.

6.22.4.8 predict() [1/3]

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

Making prediction by binding the nput data to the place_holder_ and evaluating expression_.

Parameters

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

Returns

The result this model predicts.

Example to predict

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

6.22.4.9 predict() [2/3]

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

Making prediction by binding the nput data to the place_holder_ and evaluating expression_.

Parameters

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

Returns

The result this model predicts.

Example to predict

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

6.22.4.10 predict() [3/3]

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

6.22.4.11 save_weights()

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

Writes all variables to a file

6.22.4.12 summary()

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

Print the model summary to console or to a file.

Parameters

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

6.22.4.13 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 input_layer_

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

6.22.5.3 output_layer_

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

6.22.5.4 place_holder_

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

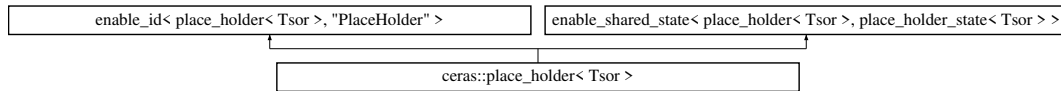
The documentation for this struct was generated from the following files:

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

6.23 ceras::place_holder< Tsr > Struct Template Reference

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

Inheritance diagram for ceras::place_holder< Tsr >:



Public Types

- typedef Tsr [tensor_type](#)

Public Member Functions

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

6.23.1 Member Typedef Documentation

6.23.1.1 tensor_type

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

6.23.2 Constructor & Destructor Documentation

6.23.2.1 place_holder() [1/4]

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

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_
```

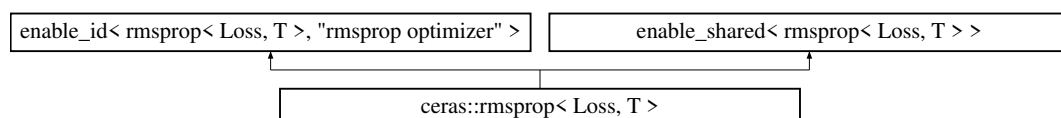
The documentation for this struct was generated from the following file:

- /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

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_
```

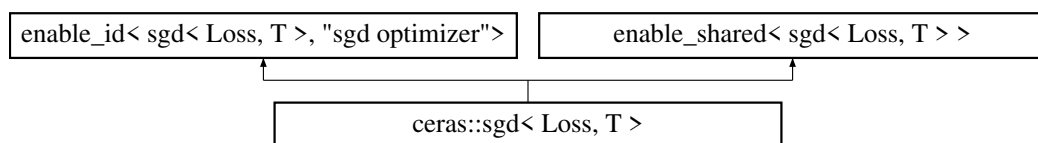
The documentation for this struct was generated from the following file:

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

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_
```

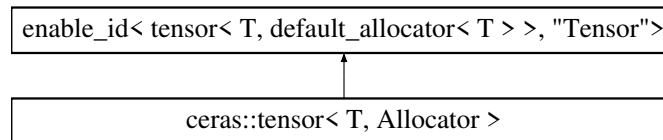
The documentation for this struct was generated from the following file:

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

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

- constexpr auto [begin](#) () noexcept
- constexpr auto [begin](#) () const noexcept
- constexpr auto [cbegin](#) () const noexcept
- constexpr auto [end](#) () noexcept
- constexpr auto [end](#) () const noexcept
- constexpr auto [cend](#) () const noexcept
- constexpr [self_type](#) & [reset](#) (T val=T{0})
- constexpr unsigned long [ndim](#) () const noexcept
- constexpr [self_type](#) & [deep_copy](#) ([self_type](#) const &other)
- constexpr [self_type](#) const [deep_copy](#) () const
- constexpr [self_type](#) const [copy](#) () const
- constexpr [value_type](#) & [operator\[\]](#) (unsigned long idx)
- constexpr [value_type](#) const & [operator\[\]](#) (unsigned long idx) const
- [tensor](#) ()
- constexpr [tensor](#) (std::vector< unsigned long > const &[shape](#), std::initializer_list< T > init, const Allocator &alloc=Allocator())
- constexpr [tensor](#) (std::vector< unsigned long > const &[shape](#))
- constexpr [tensor](#) (std::vector< unsigned long > const &[shape](#), T init)
- constexpr [tensor](#) ([tensor](#) const &other, unsigned long memory_offset)
- constexpr [tensor](#) ([self_type](#) const &other) noexcept
- constexpr [tensor](#) ([self_type](#) &&other) noexcept
- constexpr [self_type](#) & [operator=](#) ([self_type](#) const &other) noexcept
- constexpr [self_type](#) & [operator=](#) ([self_type](#) &&other) noexcept
- constexpr std::vector< unsigned long > const & [shape](#) () const noexcept
- constexpr unsigned long [size](#) () const noexcept
- constexpr [self_type](#) & [resize](#) (std::vector< unsigned long > const &new_shape)
- constexpr [self_type](#) & [reshape](#) (std::vector< unsigned long > const &new_shape)
- constexpr [self_type](#) & [shrink_to](#) (std::vector< unsigned long > const &new_shape)
- constexpr [self_type](#) & [creep_to](#) (unsigned long new_memory_offset)
- constexpr bool [empty](#) () const noexcept

- constexpr [value_type](#) * [data](#) () noexcept
- constexpr const [value_type](#) * [data](#) () const noexcept
- template<typename Function >
constexpr [self_type](#) & [map](#) (Function const &f)
- constexpr [self_type](#) & [operator+=](#) ([self_type](#) const &other)
- constexpr [self_type](#) & [operator+=](#) ([value_type](#) x)
- constexpr [self_type](#) & [operator-=](#) ([self_type](#) const &other)
- constexpr [self_type](#) & [operator-=](#) ([value_type](#) x)
- constexpr [self_type](#) & [operator*=](#) ([self_type](#) const &other)
- constexpr [self_type](#) & [operator*=](#) ([value_type](#) x)
- constexpr [self_type](#) & [operator/=](#) ([self_type](#) const &other)
- constexpr [self_type](#) & [operator/=](#) ([value_type](#) x)
- constexpr [self_type](#) const [operator-](#) () const
- constexpr [value_type as_scalar](#) () const noexcept
- template<typename U >
constexpr auto [as_type](#) () const noexcept
- [tensor slice](#) (unsigned long m, unsigned long n) const noexcept

Public Attributes

- std::vector< unsigned long > [shape_](#)
- unsigned long [memory_offset_](#)
- [shared_vector](#) [vector_](#)

6.29.1 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
```

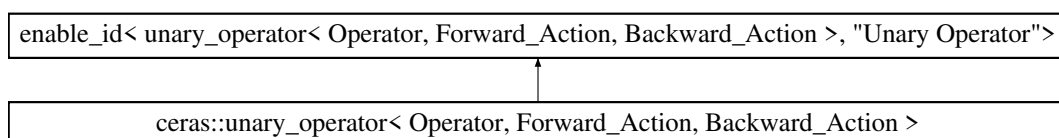
The documentation for this struct was generated from the following file:

- /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
```

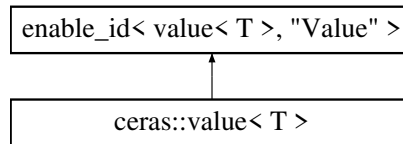
The documentation for this struct was generated from the following file:

- [/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_
```

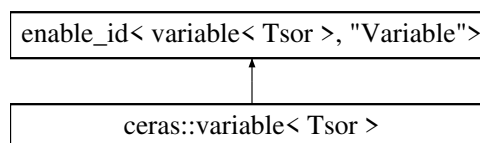
The documentation for this struct was generated from the following file:

- [/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 Types

- typedef T [value_type](#)
- typedef [value_type](#) * [row_type](#)
- typedef const [value_type](#) * [const_row_type](#)
- typedef stride_iterator< [value_type](#) * > [col_type](#)
- typedef stride_iterator< const [value_type](#) * > [const_col_type](#)

Public Member Functions

- template<typename A >
constexpr [view_2d](#) ([tensor](#)< T, A > &tsor, unsigned long [row](#), unsigned long [col](#), bool transposed=false) noexcept
- constexpr [view_2d](#) (T *[data](#), unsigned long [row](#), unsigned long [col](#), bool transposed=false) noexcept
- constexpr [view_2d](#) (const T *[data](#), unsigned long [row](#), unsigned long [col](#), bool transposed=false) noexcept
- constexpr T * [operator\[\]](#) (unsigned long index)
- constexpr const T * [operator\[\]](#) (unsigned long index) const
- constexpr auto [shape](#) () const noexcept
- constexpr unsigned long [size](#) () const noexcept
- constexpr T * [data](#) () noexcept
- constexpr const T * [data](#) () const noexcept
- constexpr T * [begin](#) () noexcept
- constexpr const T * [end](#) () const noexcept
- constexpr unsigned long [row](#) () const noexcept
- constexpr unsigned long [col](#) () const noexcept
- constexpr [row_type](#) [row_begin](#) (unsigned long index=0) noexcept
- constexpr [row_type](#) [row_end](#) (unsigned long index=0) noexcept
- constexpr [const_row_type](#) [row_begin](#) (unsigned long index=0) const noexcept
- constexpr [const_row_type](#) [row_end](#) (unsigned long index=0) const noexcept
- constexpr [col_type](#) [col_begin](#) (unsigned long index=0) noexcept
- constexpr [col_type](#) [col_end](#) (unsigned long index=0) noexcept
- constexpr [const_col_type](#) [col_begin](#) (unsigned long index=0) const noexcept
- constexpr [const_col_type](#) [col_end](#) (unsigned long index=0) const noexcept

Public Attributes

- T * [data_](#)
- unsigned long [row_](#)
- unsigned long [col_](#)
- bool [transposed_](#)

6.35.1 Member Typedef Documentation

6.35.1.1 col_type

```
template<typename T >  
typedef stride_iterator<value_type*> ceras::view_2d< T >::col_type
```

6.35.1.2 const_col_type

```
template<typename T >  
typedef stride_iterator<const value_type*> ceras::view_2d< T >::const_col_type
```

6.35.1.3 const_row_type

```
template<typename T >  
typedef const value_type* ceras::view_2d< T >::const_row_type
```

6.35.1.4 row_type

```
template<typename T >  
typedef value_type* ceras::view_2d< T >::row_type
```

6.35.1.5 value_type

```
template<typename T >  
typedef T ceras::view_2d< T >::value_type
```

6.35.2 Constructor & Destructor Documentation

6.35.2.1 `view_2d()` [1/3]

```
template<typename T >
template<typename A >
constexpr ceras::view_2d< T >::view_2d (
    tensor< T, A > & tsor,
    unsigned long row,
    unsigned long col,
    bool transposed = false ) [inline], [constexpr], [noexcept]
```

6.35.2.2 `view_2d()` [2/3]

```
template<typename T >
constexpr ceras::view_2d< T >::view_2d (
    T * data,
    unsigned long row,
    unsigned long col,
    bool transposed = false ) [inline], [constexpr], [noexcept]
```

6.35.2.3 `view_2d()` [3/3]

```
template<typename T >
constexpr ceras::view_2d< T >::view_2d (
    const T * data,
    unsigned long row,
    unsigned long col,
    bool transposed = false ) [inline], [constexpr], [noexcept]
```

6.35.3 Member Function Documentation**6.35.3.1 `begin()`**

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

6.35.3.2 `col()`

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

6.35.3.3 col_begin() [1/2]

```
template<typename T >
constexpr const_col_type ceras::view_2d< T >::col_begin (
    unsigned long index = 0 ) const [inline], [constexpr], [noexcept]
```

6.35.3.4 col_begin() [2/2]

```
template<typename T >
constexpr col_type ceras::view_2d< T >::col_begin (
    unsigned long index = 0 ) [inline], [constexpr], [noexcept]
```

6.35.3.5 col_end() [1/2]

```
template<typename T >
constexpr const_col_type ceras::view_2d< T >::col_end (
    unsigned long index = 0 ) const [inline], [constexpr], [noexcept]
```

6.35.3.6 col_end() [2/2]

```
template<typename T >
constexpr col_type ceras::view_2d< T >::col_end (
    unsigned long index = 0 ) [inline], [constexpr], [noexcept]
```

6.35.3.7 data() [1/2]

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

6.35.3.8 data() [2/2]

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

6.35.3.9 end()

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

6.35.3.10 operator[]() [1/2]

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

6.35.3.11 operator[]() [2/2]

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

6.35.3.12 row()

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

6.35.3.13 row_begin() [1/2]

```
template<typename T >
constexpr const_row_type ceras::view_2d< T >::row_begin (
    unsigned long index = 0 ) const [inline], [constexpr], [noexcept]
```

6.35.3.14 row_begin() [2/2]

```
template<typename T >
constexpr row_type ceras::view_2d< T >::row_begin (
    unsigned long index = 0 ) [inline], [constexpr], [noexcept]
```

6.35.3.15 row_end() [1/2]

```
template<typename T >
constexpr const_row_type ceras::view_2d< T >::row_end (
    unsigned long index = 0 ) const [inline], [constexpr], [noexcept]
```

6.35.3.16 row_end() [2/2]

```
template<typename T >
constexpr row_type ceras::view_2d< T >::row_end (
    unsigned long index = 0 ) [inline], [constexpr], [noexcept]
```

6.35.3.17 shape()

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

6.35.3.18 size()

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

6.35.4 Member Data Documentation

6.35.4.1 col_

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

6.35.4.2 data_

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

6.35.4.3 row_

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

6.35.4.4 transposed_

```
template<typename T >
bool ceras::view_2d< T >::transposed_
```

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

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

6.36 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).
- auto [ceras::repeat](#) (unsigned long repeats, unsigned long axis=-1) noexcept
Repeats elements along an axis.
- auto [ceras::reduce_min](#) (unsigned long axis=-1) noexcept
Reduce minimal elements along an axis.
- auto [ceras::reduce_max](#) (unsigned long axis=-1) noexcept
Reduce maximum elements along an axis.

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< T_{src} >](#)

Namespaces

- [ceras](#)
- [ceras::ceras_private](#)

Functions

- template<Tensor T_{src}>
[ceras_private::session< T_{src} > & 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 Tsor>`
`Tsor ceras::operator- (typename Tsor::value_type const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator- (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator* (typename Tsor::value_type const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator* (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator/ (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::reshape (Tsor const &ts, std::vector< unsigned long > const &new_shape)`
- `template<Tensor Tsor>`
`void ceras::multiply (Tsor const &lhs, Tsor const &rhs, Tsor &ans) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::multiply (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator* (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::elementwise_product (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::hadamard_product (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::elementwise_divide (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::repeat (Tsor const &tsor, unsigned long n)`
- `template<Tensor Tsor>`
`Tsor ceras::reduce_sum (Tsor const &tsor)`
- `template<Tensor Tsor>`
`Tsor ceras::reduce_mean (Tsor const &tsor)`
- `template<Tensor Tsor>`
`Tsor ceras::clip (Tsor &tsor, typename Tsor::value_type lower=0, typename Tsor::value_type upper=1)`
- `template<Tensor Tsor>`
`Tsor ceras::squeeze (Tsor const &tsor)`
- `template<typename T , typename A = default_allocator<T>>>`
`tensor< T, A > ceras::randn (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1})`
- `template<typename T , typename A = default_allocator<T>>>`
`tensor< T, A > ceras::truncated_normal (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1}, T lower=T{0}, T upper=T{1})`
- `template<typename T , typename A = default_allocator<T>>>`
`tensor< T, A > ceras::random (std::vector< unsigned long > const &shape, T min=T{0}, T max=T{1})`
- `template<Tensor 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 Tsor>`
`auto ceras::max (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto ceras::amax (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto ceras::min (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto ceras::amin (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto ceras::sum (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto ceras::mean (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto ceras::norm (Tsor const &tsor)`
- `template<Tensor Tsor>`
`Tsor ceras::abs (Tsor const &tsor)`
- `template<Tensor Tsor>`
`Tsor ceras::softmax (Tsor const &tsor)`
- `template<Tensor Tsor>`
`bool ceras::has_nan (Tsor const &tsor)`
- `template<Tensor Tsor>`
`bool ceras::has_inf (Tsor const &tsor)`
- `template<Tensor Tsor>`
`bool ceras::is_valid (Tsor const &tsor)`
- `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`
- `template<Tensor Tsor>`
`Tsor ceras::sum (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`
`requires std::floating_point< typename Tsor::value_type > Tsor ceras::mean (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`
`requires std::floating_point< typename Tsor::value_type > Tsor ceras::variance (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`
`requires std::floating_point< typename Tsor::value_type > Tsor ceras::standard_deviation (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`
`requires std::floating_point< typename Tsor::value_type > Tsor::value_type ceras::var (Tsor const &ts) noexcept`
- `template<Tensor Tsor>`
`requires std::floating_point< typename Tsor::value_type > Tsor::value_type ceras::std (Tsor const &ts) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::max (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`

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

Variables

- static unsigned long [ceras::random_seed](#) = std::chrono::system_clock::now().time_since_epoch().count()
- static std::mt19937 [ceras::random_generator](#) {random_seed}
- template<class T >
constexpr bool [ceras::is_tensor_v](#) = is_tensor<T>::value
- template<typename T >
concept [ceras::Tensor](#) = is_tensor_v<T>

7.15 /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>`

7.17 /data/structured_folders/workspace/github.repo/ceras/include/xmodel.hpp File Reference

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

Classes

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

Namespaces

- [ceras](#)

Index

/data/structured_folders/workspace/github.repo/ceras/include/activation.hpp, 58
139
adagrad
/data/structured_folders/workspace/github.repo/ceras/include/ceras/adagrad< Loss, T >, 66
140
Adam
/data/structured_folders/workspace/github.repo/ceras/include/config.hpp, 58
140
adam
/data/structured_folders/workspace/github.repo/ceras/include/ceras/adam< Loss, T >, 68
140
Add
/data/structured_folders/workspace/github.repo/ceras/include/data.hpp, 19
141
add
/data/structured_folders/workspace/github.repo/ceras/include/include.hpp, 19
142
allocator
/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp, 107
143
amax
/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp, 20
144
amin
/data/structured_folders/workspace/github.repo/ceras/include/model.hpp, 20
145
amsgrad
/data/structured_folders/workspace/github.repo/ceras/include/operator.hpp< Loss, T >, 68
146
as_scalar
/data/structured_folders/workspace/github.repo/ceras/include/operator.hpp< T, Allocator >, 109
149
as_tensor
/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp, 20
150
as_type
/data/structured_folders/workspace/github.repo/ceras/include/serializer.hpp< T, Allocator >, 109
151
average_pooling_2d
/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp, 20
151
AveragePooling2D
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp, 20
155
backward
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp, 20
156
ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >, 71
/data/structured_folders/workspace/github.repo/ceras/include/xmodel.hpp, 20
157
ceras::constant< Tzor >, 78
ceras::place_holder< Tzor >, 93
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 117
ceras::value< T >, 120
ceras::variable< Tzor >, 123
backward_action_
ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >, 71
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 118
batch_normalization
ceras, 20
batch_size_
ceras::view_4d< T >, 138
BatchNormalization
ceras, 21
~session
ceras::ceras_private::session< Tzor >, 100
abs
ceras, 19
abs_loss
ceras, 19
ada_delta
ceras, 18
ada_grad
ceras, 18
Adadelta
ceras, 53
adadelta
ceras::adadelta< Loss, T >, 64
Adagrad

begin
 ceras::tensor< T, Allocator >, 109
 ceras::view_2d< T >, 129
 beta_1_
 ceras::adam< Loss, T >, 68
 beta_2_
 ceras::adam< Loss, T >, 69
 binary_cross_entropy_loss
 ceras, 21
 Binary_Operator
 ceras, 53
 binary_operator
 ceras::binary_operator< Lhs_Operator, Rhs_Operator,
 Forward_Action, Backward_Action >, 70
 BinaryCrossEntropy
 ceras, 54
 BinaryCrossentropy
 ceras, 54
 bind
 ceras::ceras_private::session< Tsor >, 101
 ceras::place_holder< Tsor >, 93

 CategoricalCrossEntropy
 ceras, 54
 CategoricalCrossentropy
 ceras, 54
 cbegin
 ceras::tensor< T, Allocator >, 110
 cend
 ceras::tensor< T, Allocator >, 110
 ceras, 9
 abs, 19
 abs_loss, 19
 ada_delta, 18
 ada_grad, 18
 Adadelta, 53
 Adagrad, 53
 Adam, 53
 Add, 19
 add, 19
 amax, 20
 amin, 20
 as_tensor, 20
 average_pooling_2d, 20
 AveragePooling2D, 20
 batch_normalization, 20
 BatchNormalization, 21
 binary_cross_entropy_loss, 21
 Binary_Operator, 53
 BinaryCrossEntropy, 54
 BinaryCrossentropy, 54
 CategoricalCrossEntropy, 54
 CategoricalCrossentropy, 54
 clip, 22
 computation_graph, 22
 concat, 22
 Concatenate, 23
 concatenate, 23
 Constant, 55
 Conv2D, 23
 conv2d, 24
 copy, 24
 cross_entropy, 25
 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, 27
 ELU, 27
 elu, 27
 empty, 27
 equal, 27
 exp, 28
 exponential, 28
 Expression, 55
 Flatten, 28
 flatten, 28
 gelu, 28
 gemm, 29
 gemm_cpu, 29
 get_default_session, 29
 glorot_uniform, 29
 hadamard_product, 30
 hard_sigmoid, 30
 has_inf, 30
 has_nan, 30
 Hinge, 55
 hinge_loss, 30
 identity, 31
 img2col, 31
 Input, 31
 is_binary_operator_v, 55
 is_constant_v, 55
 is_place_holder_v, 56
 is_tensor_v, 56
 is_unary_operator_v, 56
 is_valid, 31
 is_value_v, 56
 is_variable_v, 56
 leaky_relu, 31
 LeakyReLU, 31
 linspace, 32
 load_tensor, 32
 log, 32
 MAE, 56
 mae, 32
 make_binary_operator, 57
 make_compiled_model, 32
 make_trainable, 32
 make_unary_operator, 57
 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, 57
MeanSquaredError, 57
min, 35
minus, 35
MSE, 58
mse, 35
Multiply, 36
multiply, 36
negative, 36
negative_relu, 36
norm, 37
normalization_batch, 37
ones, 37
ones_like, 37
Operator, 58
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, 58
plus, 41
randn, 41
randn_like, 42
random, 42
random_generator, 58
random_like, 42
random_normal_like, 42
random_seed, 59
read_tensor, 43
reduce, 43
reduce_max, 43
reduce_mean, 44
reduce_min, 44
reduce_sum, 44
ReLU, 45
relu, 45
repeat, 45
replace_placeholder_with_expression, 46
repmat, 46
Reshape, 46
reshape, 46
rms_prop, 18
RMSprop, 59
save_tensor, 47
selu, 47
SGD, 59
sigmoid, 47
sign, 47
Softmax, 48
softmax, 48
softplus, 48
softsign, 48
square, 48
squared_loss, 49
squeeze, 49
standard_deviation, 49
std, 49
Subtract, 49
sum, 50
sum_reduce, 50
tanh, 50
Tensor, 59
tesseract, 18
transpose, 50
truncated_normal, 51
Unary_Operator, 59
up_sampling_2d, 51
update_cuda_gemm_threshold, 51
UpSampling2D, 51
Value, 60
var, 51
Variable, 60
variance, 51
write_tensor, 52
zero_padding_2d, 52
zeros, 52
zeros_like, 52
ceras::adadelta< Loss, T >, 63
 adadelta, 64
 forward, 64
 iterations_, 64
 learning_rate_, 64
 loss_, 64
 rho_, 65
 tensor_type, 63
ceras::adagrad< Loss, T >, 65
 adagrad, 66
 decay_, 66
 forward, 66
 iterations_, 66
 learning_rate_, 66
 loss_, 67
 tensor_type, 65
ceras::adam< Loss, T >, 67
 adam, 68
 amsgrad_, 68
 beta_1_, 68
 beta_2_, 69
 forward, 68
 iterations_, 69
 learning_rate_, 69

- loss_, 69
- tensor_type, 68
- ceras::binary_operator< Lhs_Operator, Rhs_Operator,
Forward_Action, Backward_Action >, 69
 - backward, 71
 - backward_action_, 71
 - binary_operator, 70
 - forward, 71
 - forward_action_, 71
 - lhs_input_data_, 71
 - lhs_op_, 71
 - output_data_, 72
 - rhs_input_data_, 72
 - rhs_op_, 72
 - tensor_type, 70
- ceras::ceras_private, 60
- ceras::ceras_private::session< Tzor >, 99
 - ~session, 100
 - bind, 101
 - deserialize, 101
 - operator=, 101
 - place_holder_type, 99
 - place_holders_, 103
 - rebind, 101
 - remember, 101
 - restore, 102
 - run, 102
 - save, 102
 - serialize, 102
 - session, 100
 - tap, 102
 - variable_state_type, 99
 - variable_type, 100
 - variables_, 103
- ceras::compiled_model< Model, Optimizer, Loss >, 72
 - compiled_model, 73
 - compiled_optimizer_, 76
 - evaluate, 74
 - fit, 74
 - ground_truth_place_holder_, 76
 - input_place_holder_, 76
 - io_layer_type, 73
 - loss_, 76
 - model_, 76
 - operator(), 75
 - optimizer_, 77
 - optimizer_type, 77
 - predict, 75
 - train_on_batch, 75
 - trainable, 76
- ceras::constant< Tzor >, 77
 - backward, 78
 - constant, 78
 - data_, 78
 - forward, 78
 - shape, 78
- ceras::dataset, 60
- ceras::dataset::fashion_mnist, 60
- load_data, 60
- ceras::dataset::mnist, 61
 - load_data, 61
- ceras::gradient_descent< Loss, T >, 79
 - forward, 80
 - gradient_descent, 79
 - learning_rate_, 80
 - loss_, 80
 - momentum_, 80
 - tensor_type, 79
- ceras::is_binary_operator< binary_operator< Lhs_Operator,
Rhs_Operator, Forward_Action, Back-
ward_Action > >, 81
- ceras::is_binary_operator< T >, 80
- ceras::is_constant< constant< Tzor > >, 81
- ceras::is_constant< T >, 81
- ceras::is_place_holder< place_holder< Tzor > >, 82
- ceras::is_place_holder< T >, 82
- ceras::is_tensor< T >, 82
- ceras::is_tensor< tensor< T, A > >, 83
- ceras::is_unary_operator< T >, 83
- ceras::is_unary_operator< unary_operator< Operator,
Forward_Action, Backward_Action > >, 83
- ceras::is_value< T >, 84
- ceras::is_value< value< T > >, 84
- ceras::is_variable< T >, 84
- ceras::is_variable< variable< Tzor > >, 85
- ceras::model< Ex, Ph >, 85
 - compile, 87
 - expression_, 91
 - input, 87, 88
 - input_layer_, 91
 - input_layer_type, 86
 - load_weights, 88
 - model, 86, 87
 - operator(), 88
 - output, 88, 89
 - output_layer_, 91
 - output_layer_type, 86
 - place_holder_, 91
 - predict, 89, 90
 - save_weights, 90
 - summary, 90
 - trainable, 90
- ceras::place_holder< Tzor >, 92
 - backward, 93
 - bind, 93
 - forward, 93
 - operator=, 93, 94
 - place_holder, 92, 93
 - reset, 94
 - tensor_type, 92
- ceras::place_holder_state< Tzor >, 94
 - data_, 94
 - shape_hint_, 94
- ceras::regularizer< Float >, 95
 - l1_, 96
 - l2_, 96

- regularizer, 95
- synchronized_, 96
- value_type, 95
- ceras::rmsprop< Loss, T >, 96
 - decay_, 98
 - forward, 97
 - iterations_, 98
 - learning_rate_, 98
 - loss_, 98
 - rho_, 98
 - rmsprop, 97
 - tensor_type, 97
- ceras::sgd< Loss, T >, 103
 - decay_, 104
 - forward, 104
 - iterations_, 105
 - learning_rate_, 105
 - loss_, 105
 - momentum_, 105
 - nesterov_, 105
 - sgd, 104
 - tensor_type, 104
- ceras::tensor< T, Allocator >, 106
 - allocator, 107
 - as_scalar, 109
 - as_type, 109
 - begin, 109
 - cbegin, 110
 - cend, 110
 - copy, 110
 - creep_to, 110
 - data, 110
 - deep_copy, 110, 111
 - empty, 111
 - end, 111
 - map, 111
 - memory_offset_, 115
 - ndim, 111
 - operator*=: 112
 - operator+=, 112
 - operator-, 112
 - operator-=, 112, 113
 - operator/=: 113
 - operator=, 113
 - operator[], 113, 114
 - reset, 114
 - reshape, 114
 - resize, 114
 - self_type, 107
 - shape, 114
 - shape_, 115
 - shared_vector, 107
 - shrink_to, 115
 - size, 115
 - slice, 115
 - tensor, 108, 109
 - value_type, 107
 - vector_, 115
 - vector_type, 108
- ceras::tensor_deduction< L, R >, 116
 - op_type, 116
 - tensor_type, 116
- ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 116
 - backward, 117
 - backward_action_, 118
 - forward, 117
 - forward_action_, 118
 - input_data_, 118
 - op_, 118
 - output_data_, 118
 - tensor_type, 118
 - unary_operator, 117
- ceras::value< T >, 119
 - backward, 120
 - data_, 121
 - forward, 120
 - operator=, 120, 121
 - value, 119, 120
 - value_type, 119
- ceras::variable< Tsor >, 121
 - backward, 123
 - contexts, 123
 - data, 124
 - forward, 124
 - gradient, 124
 - operator=, 124
 - regularizer_, 125
 - reset, 125
 - shape, 125
 - state_, 125
 - tensor_type, 122
 - trainable, 125
 - trainable_, 126
 - value_type, 122
 - variable, 122, 123
- ceras::variable_state< Tsor >, 126
 - contexts_, 126
 - data_, 126
 - gradient_, 126
- ceras::view_2d< T >, 127
 - begin, 129
 - col, 129
 - col_, 132
 - col_begin, 129, 130
 - col_end, 130
 - col_type, 128
 - const_col_type, 128
 - const_row_type, 128
 - data, 130
 - data_, 132
 - end, 130
 - operator[], 131
 - row, 131
 - row_, 132
 - row_begin, 131

- row_end, [131](#), [132](#)
- row_type, [128](#)
- shape, [132](#)
- size, [132](#)
- transposed_, [133](#)
- value_type, [128](#)
- view_2d, [128](#), [129](#)
- ceras::view_3d< T >, [133](#)
 - channel_, [134](#)
 - col_, [134](#)
 - data_, [134](#)
 - operator[], [134](#)
 - row_, [134](#)
 - view_3d, [133](#)
- ceras::view_4d< T >, [135](#)
 - batch_size_, [138](#)
 - channel_, [138](#)
 - col_, [138](#)
 - data_, [138](#)
 - operator[], [137](#)
 - row_, [138](#)
 - view_4d, [135](#)
- channel_
 - ceras::view_3d< T >, [134](#)
 - ceras::view_4d< T >, [138](#)
- clip
 - ceras, [22](#)
- col
 - ceras::view_2d< T >, [129](#)
- col_
 - ceras::view_2d< T >, [132](#)
 - ceras::view_3d< T >, [134](#)
 - ceras::view_4d< T >, [138](#)
- col_begin
 - ceras::view_2d< T >, [129](#), [130](#)
- col_end
 - ceras::view_2d< T >, [130](#)
- col_type
 - ceras::view_2d< T >, [128](#)
- compile
 - ceras::model< Ex, Ph >, [87](#)
- compiled_model
 - ceras::compiled_model< Model, Optimizer, Loss >, [73](#)
- compiled_optimizer_
 - ceras::compiled_model< Model, Optimizer, Loss >, [76](#)
- computation_graph
 - ceras, [22](#)
- concat
 - ceras, [22](#)
- Concatenate
 - ceras, [23](#)
- concatenate
 - ceras, [23](#)
- const_col_type
 - ceras::view_2d< T >, [128](#)
- const_row_type
 - ceras::view_2d< T >, [128](#)
- Constant
 - ceras, [55](#)
- constant
 - ceras::constant< Tzor >, [78](#)
- contexts
 - ceras::variable< Tzor >, [123](#)
- contexts_
 - ceras::variable_state< Tzor >, [126](#)
- Conv2D
 - ceras, [23](#)
- conv2d
 - ceras, [24](#)
- copy
 - ceras, [24](#)
 - ceras::tensor< T, Allocator >, [110](#)
- creep_to
 - ceras::tensor< T, Allocator >, [110](#)
- cross_entropy
 - ceras, [25](#)
- cross_entropy_loss
 - ceras, [25](#)
- cube
 - ceras, [18](#)
- data
 - ceras::tensor< T, Allocator >, [110](#)
 - ceras::variable< Tzor >, [124](#)
 - ceras::view_2d< T >, [130](#)
- data_
 - ceras::constant< Tzor >, [78](#)
 - ceras::place_holder_state< Tzor >, [94](#)
 - ceras::value< T >, [121](#)
 - ceras::variable_state< Tzor >, [126](#)
 - ceras::view_2d< T >, [132](#)
 - ceras::view_3d< T >, [134](#)
 - ceras::view_4d< T >, [138](#)
- decay_
 - ceras::adagrad< Loss, T >, [66](#)
 - ceras::rmsprop< Loss, T >, [98](#)
 - ceras::sgd< Loss, T >, [104](#)
- deep_copy
 - ceras, [25](#)
 - ceras::tensor< T, Allocator >, [110](#), [111](#)
- default_allocator
 - ceras, [18](#)
- Dense
 - ceras, [25](#)
- deserialize
 - ceras::ceras_private::session< Tzor >, [101](#)
- drop_out
 - ceras, [26](#)
- Dropout
 - ceras, [26](#)
- elementwise_divide
 - ceras, [26](#)
- elementwise_multiply
 - ceras, [26](#)

- elementwise_product
 - ceras, [26, 27](#)
- ELU
 - ceras, [27](#)
- elu
 - ceras, [27](#)
- empty
 - ceras, [27](#)
 - ceras::tensor< T, Allocator >, [111](#)
- end
 - ceras::tensor< T, Allocator >, [111](#)
 - ceras::view_2d< T >, [130](#)
- equal
 - ceras, [27](#)
- evaluate
 - ceras::compiled_model< Model, Optimizer, Loss >, [74](#)
- exp
 - ceras, [28](#)
- exponential
 - ceras, [28](#)
- Expression
 - ceras, [55](#)
- expression_
 - ceras::model< Ex, Ph >, [91](#)
- fit
 - ceras::compiled_model< Model, Optimizer, Loss >, [74](#)
- Flatten
 - ceras, [28](#)
- flatten
 - ceras, [28](#)
- forward
 - ceras::adadelta< Loss, T >, [64](#)
 - ceras::adagrad< Loss, T >, [66](#)
 - ceras::adam< Loss, T >, [68](#)
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [71](#)
 - ceras::constant< Tsor >, [78](#)
 - ceras::gradient_descent< Loss, T >, [80](#)
 - ceras::place_holder< Tsor >, [93](#)
 - ceras::rmsprop< Loss, T >, [97](#)
 - ceras::sgd< Loss, T >, [104](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [117](#)
 - ceras::value< T >, [120](#)
 - ceras::variable< Tsor >, [124](#)
- forward_action_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [71](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [118](#)
- gelu
 - ceras, [28](#)
- gemm
 - ceras, [29](#)
- gemm_cpu
 - ceras, [29](#)
- get_default_session
 - ceras, [29](#)
- glorot_uniform
 - ceras, [29](#)
- gradient
 - ceras::variable< Tsor >, [124](#)
- gradient_
 - ceras::variable_state< Tsor >, [126](#)
- gradient_descent
 - ceras::gradient_descent< Loss, T >, [79](#)
- ground_truth_place_holder_
 - ceras::compiled_model< Model, Optimizer, Loss >, [76](#)
- hadamard_product
 - ceras, [30](#)
- hard_sigmoid
 - ceras, [30](#)
- has_inf
 - ceras, [30](#)
- has_nan
 - ceras, [30](#)
- Hinge
 - ceras, [55](#)
- hinge_loss
 - ceras, [30](#)
- identity
 - ceras, [31](#)
- img2col
 - ceras, [31](#)
- includes.hpp
 - STB_IMAGE_IMPLEMENTATION, [142](#)
 - STB_IMAGE_RESIZE_IMPLEMENTATION, [143](#)
 - STB_IMAGE_WRITE_IMPLEMENTATION, [143](#)
- Input
 - ceras, [31](#)
- input
 - ceras::model< Ex, Ph >, [87, 88](#)
- input_data_
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [118](#)
- input_layer_
 - ceras::model< Ex, Ph >, [91](#)
- input_layer_type
 - ceras::model< Ex, Ph >, [86](#)
- input_place_holder_
 - ceras::compiled_model< Model, Optimizer, Loss >, [76](#)
- io_layer_type
 - ceras::compiled_model< Model, Optimizer, Loss >, [73](#)
- is_binary_operator_v
 - ceras, [55](#)
- is_constant_v
 - ceras, [55](#)
- is_place_holder_v
 - ceras, [56](#)

- is_tensor_v
 - ceras, 56
- is_unary_operator_v
 - ceras, 56
- is_valid
 - ceras, 31
- is_value_v
 - ceras, 56
- is_variable_v
 - ceras, 56
- iterations_
 - ceras::adadelta< Loss, T >, 64
 - ceras::adagrad< Loss, T >, 66
 - ceras::adam< Loss, T >, 69
 - ceras::rmsprop< Loss, T >, 98
 - ceras::sgd< Loss, T >, 105
- l1_
 - ceras::regularizer< Float >, 96
- l2_
 - ceras::regularizer< Float >, 96
- leaky_relu
 - ceras, 31
- LeakyReLU
 - ceras, 31
- learning_rate_
 - ceras::adadelta< Loss, T >, 64
 - ceras::adagrad< Loss, T >, 66
 - ceras::adam< Loss, T >, 69
 - ceras::gradient_descent< Loss, T >, 80
 - ceras::rmsprop< Loss, T >, 98
 - ceras::sgd< Loss, T >, 105
- lhs_input_data_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 71
- lhs_op_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 71
- linspace
 - ceras, 32
- load_data
 - ceras::dataset::fashion_mnist, 60
 - ceras::dataset::mnist, 61
- load_tensor
 - ceras, 32
- load_weights
 - ceras::model< Ex, Ph >, 88
- log
 - ceras, 32
- loss_
 - ceras::adadelta< Loss, T >, 64
 - ceras::adagrad< Loss, T >, 67
 - ceras::adam< Loss, T >, 69
 - ceras::compiled_model< Model, Optimizer, Loss >, 76
 - ceras::gradient_descent< Loss, T >, 80
 - ceras::rmsprop< Loss, T >, 98
 - ceras::sgd< Loss, T >, 105
- MAE
 - ceras, 56
- mae
 - ceras, 32
- make_binary_operator
 - ceras, 57
- make_compiled_model
 - ceras, 32
- make_trainable
 - ceras, 32
- make_unary_operator
 - ceras, 57
- map
 - ceras::tensor< T, Allocator >, 111
- matrix
 - ceras, 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, 57
- MeanSquaredError
 - ceras, 57
- memory_offset_
 - ceras::tensor< T, Allocator >, 115
- min
 - ceras, 35
- minus
 - ceras, 35
- model
 - ceras::model< Ex, Ph >, 86, 87
- model_
 - ceras::compiled_model< Model, Optimizer, Loss >, 76
- momentum_
 - ceras::gradient_descent< Loss, T >, 80
 - ceras::sgd< Loss, T >, 105
- MSE
 - ceras, 58
- mse
 - ceras, 35
- Multiply
 - ceras, 36

multiply
 ceras, 36

ndim
 ceras::tensor< T, Allocator >, 111

negative
 ceras, 36

negative_relu
 ceras, 36

nesterov_
 ceras::sgd< Loss, T >, 105

norm
 ceras, 37

normalization_batch
 ceras, 37

ones
 ceras, 37

ones_like
 ceras, 37

op_
 ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 118

op_type
 ceras::tensor_deduction< L, R >, 116

Operator
 ceras, 58

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 >, 112

operator()
 ceras::compiled_model< Model, Optimizer, Loss >, 75
 ceras::model< Ex, Ph >, 88

operator+
 ceras, 38, 39

operator+=
 ceras::tensor< T, Allocator >, 112

operator-
 ceras, 39, 40
 ceras::tensor< T, Allocator >, 112

operator-=
 ceras::tensor< T, Allocator >, 112, 113

operator/
 ceras, 40

operator/=
 ceras::tensor< T, Allocator >, 113

operator=
 ceras::ceras_private::session< Tsor >, 101
 ceras::place_holder< Tsor >, 93, 94
 ceras::tensor< T, Allocator >, 113
 ceras::value< T >, 120, 121
 ceras::variable< Tsor >, 124

operator==
 ceras, 40, 41

operator[]
 ceras::tensor< T, Allocator >, 113, 114
 ceras::view_2d< T >, 131
 ceras::view_3d< T >, 134
 ceras::view_4d< T >, 137

optimizer_
 ceras::compiled_model< Model, Optimizer, Loss >, 77

optimizer_type
 ceras::compiled_model< Model, Optimizer, Loss >, 77

output
 ceras::model< Ex, Ph >, 88, 89

output_data_
 ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 72
 ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 118

output_layer_
 ceras::model< Ex, Ph >, 91

output_layer_type
 ceras::model< Ex, Ph >, 86

Place_Holder
 ceras, 58

place_holder
 ceras::place_holder< Tsor >, 92, 93

place_holder_
 ceras::model< Ex, Ph >, 91

place_holder_type
 ceras::ceras_private::session< Tsor >, 99

place_holders_
 ceras::ceras_private::session< Tsor >, 103

plus
 ceras, 41

predict
 ceras::compiled_model< Model, Optimizer, Loss >, 75
 ceras::model< Ex, Ph >, 89, 90

randn
 ceras, 41

randn_like
 ceras, 42

random
 ceras, 42

random_generator
 ceras, 58

random_like
 ceras, 42

random_normal_like
 ceras, 42
 random_seed
 ceras, 59
 read_tensor
 ceras, 43
 rebind
 ceras::ceras_private::session< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 101
 reduce
 ceras, 43
 reduce_max
 ceras, 43
 reduce_mean
 ceras, 44
 reduce_min
 ceras, 44
 reduce_sum
 ceras, 44
 regularizer
 ceras::regularizer< Float, S, A, V, R, O, S, I, D, E, F, C, E, S >, 95
 regularizer_
 ceras::variable< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 125
 ReLU
 ceras, 45
 relu
 ceras, 45
 remember
 ceras::ceras_private::session< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 101
 repeat
 ceras, 45
 replace_placeholder_with_expression
 ceras, 46
 repmat
 ceras, 46
 reset
 ceras::place_holder< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 94
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 114
 ceras::variable< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 125
 Reshape
 ceras, 46
 reshape
 ceras, 46
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 114
 resize
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 114
 restore
 ceras::ceras_private::session< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 102
 rho_
 ceras::adadelat< Loss, T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 65
 ceras::rmsprop< Loss, T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 98
 rhs_input_data_
 ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 72
 rhs_op_
 ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 72
 rms_prop
 ceras, 18
 RMSprop
 ceras, 59
 rmsprop
 ceras::rmsprop< Loss, T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 97
 row
 ceras::view_2d< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 131
 row_
 ceras::view_2d< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 132
 ceras::view_3d< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 134
 ceras::view_4d< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 138
 row_begin
 ceras::view_2d< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 131
 row_end
 ceras::view_2d< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 131, 132
 row_type
 ceras::view_2d< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 128
 run
 ceras::ceras_private::session< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 102
 save
 ceras::ceras_private::session< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 102
 save_tensor
 ceras, 47
 save_weights
 ceras::model< Ex, Ph, S, A, V, R, O, S, I, D, E, F, C, E, S >, 90
 self_type
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 107
 selu
 ceras, 47
 serialize
 ceras::ceras_private::session< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 102
 session
 ceras::ceras_private::session< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 100
 SGD
 ceras, 59
 sgd
 ceras::sgd< Loss, T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 104
 shape
 ceras::constant< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 78
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 114
 ceras::variable< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 125
 ceras::view_2d< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 132
 shape_
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 115
 shape_hint_
 ceras::place_holder_state< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 94
 shared_vector
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 107
 shrink_to
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 115
 sigmoid
 ceras, 47
 sign
 ceras, 47
 size
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 115
 ceras::view_2d< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 132
 slice
 ceras::tensor< T, S, A, V, R, O, S, I, D, E, F, C, E, S >, 115

- Softmax
 - ceras, [48](#)
- softmax
 - ceras, [48](#)
- softplus
 - ceras, [48](#)
- softsign
 - ceras, [48](#)
- square
 - ceras, [48](#)
- squared_loss
 - ceras, [49](#)
- squeeze
 - ceras, [49](#)
- standard_deviation
 - ceras, [49](#)
- state_
 - ceras::variable< Tsor >, [125](#)
- STB_IMAGE_IMPLEMENTATION
 - includes.hpp, [142](#)
- STB_IMAGE_RESIZE_IMPLEMENTATION
 - includes.hpp, [143](#)
- STB_IMAGE_WRITE_IMPLEMENTATION
 - includes.hpp, [143](#)
- std
 - ceras, [49](#)
- Subtract
 - ceras, [49](#)
- sum
 - ceras, [50](#)
- sum_reduce
 - ceras, [50](#)
- summary
 - ceras::model< Ex, Ph >, [90](#)
- synchronized_
 - ceras::regularizer< Float >, [96](#)
- tanh
 - ceras, [50](#)
- tap
 - ceras::ceras_private::session< Tsor >, [102](#)
- Tensor
 - ceras, [59](#)
- tensor
 - ceras::tensor< T, Allocator >, [108](#), [109](#)
- tensor_type
 - ceras::adadelta< Loss, T >, [63](#)
 - ceras::adagrad< Loss, T >, [65](#)
 - ceras::adam< Loss, T >, [68](#)
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [70](#)
 - ceras::gradient_descent< Loss, T >, [79](#)
 - ceras::place_holder< Tsor >, [92](#)
 - ceras::rmsprop< Loss, T >, [97](#)
 - ceras::sgd< Loss, T >, [104](#)
 - ceras::tensor_deduction< L, R >, [116](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [118](#)
 - ceras::variable< Tsor >, [122](#)
- tesseract
 - ceras, [18](#)
- train_on_batch
 - ceras::compiled_model< Model, Optimizer, Loss >, [75](#)
- trainable
 - ceras::compiled_model< Model, Optimizer, Loss >, [76](#)
 - ceras::model< Ex, Ph >, [90](#)
 - ceras::variable< Tsor >, [125](#)
- trainable_
 - ceras::variable< Tsor >, [126](#)
- transpose
 - ceras, [50](#)
- transposed_
 - ceras::view_2d< T >, [133](#)
- truncated_normal
 - ceras, [51](#)
- Unary_Operator
 - ceras, [59](#)
- unary_operator
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [117](#)
- up_sampling_2d
 - ceras, [51](#)
- update_cuda_gemm_threshold
 - ceras, [51](#)
- UpSampling2D
 - ceras, [51](#)
- Value
 - ceras, [60](#)
- value
 - ceras::value< T >, [119](#), [120](#)
- value_type
 - ceras::regularizer< Float >, [95](#)
 - ceras::tensor< T, Allocator >, [107](#)
 - ceras::value< T >, [119](#)
 - ceras::variable< Tsor >, [122](#)
 - ceras::view_2d< T >, [128](#)
- var
 - ceras, [51](#)
- Variable
 - ceras, [60](#)
- variable
 - ceras::variable< Tsor >, [122](#), [123](#)
- variable_state_type
 - ceras::ceras_private::session< Tsor >, [99](#)
- variable_type
 - ceras::ceras_private::session< Tsor >, [100](#)
- variables_
 - ceras::ceras_private::session< Tsor >, [103](#)
- variance
 - ceras, [51](#)
- vector_
 - ceras::tensor< T, Allocator >, [115](#)
- vector_type
 - ceras::tensor< T, Allocator >, [108](#)

view_2d
 ceras::view_2d< T >, [128](#), [129](#)
view_3d
 ceras::view_3d< T >, [133](#)
view_4d
 ceras::view_4d< T >, [135](#)

write_tensor
 ceras, [52](#)

zero_padding_2d
 ceras, [52](#)
zeros
 ceras, [52](#)
zeros_like
 ceras, [52](#)