

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	17
5.1.1.1 ada_delta	17
5.1.1.2 ada_grad	17
5.1.1.3 cube	17
5.1.1.4 default_allocator	17
5.1.1.5 matrix	18
5.1.1.6 rms_prop	18
5.1.1.7 tesseract	18
5.1.2 Function Documentation	18
5.1.2.1 abs() [1/2]	18
5.1.2.2 abs() [2/2]	18
5.1.2.3 abs_loss()	18
5.1.2.4 Add()	19
5.1.2.5 add()	19
5.1.2.6 amax()	19
5.1.2.7 amin()	19
5.1.2.8 as_tensor()	19
5.1.2.9 average_pooling_2d()	20
5.1.2.10 batch_normalization()	20
5.1.2.11 BatchNormalization()	20
5.1.2.12 clip() [1/2]	20
5.1.2.13 clip() [2/2]	20
5.1.2.14 concat() [1/2]	20
5.1.2.15 concat() [2/2]	21
5.1.2.16 concatenate() [1/3]	21
5.1.2.17 concatenate() [2/3]	21
5.1.2.18 concatenate() [3/3]	21
5.1.2.19 Concatenate()	21
5.1.2.20 Conv2D()	22
5.1.2.21 conv2d()	22

5.1.2.22 copy() [1/2]	22
5.1.2.23 copy() [2/2]	22
5.1.2.24 cross_entropy()	22
5.1.2.25 cross_entropy_loss()	23
5.1.2.26 deep_copy()	23
5.1.2.27 Dense()	23
5.1.2.28 drop_out()	23
5.1.2.29 elementwise_divide()	23
5.1.2.30 elementwise_product() [1/2]	23
5.1.2.31 elementwise_product() [2/2]	24
5.1.2.32 elu()	24
5.1.2.33 ELU()	24
5.1.2.34 empty()	24
5.1.2.35 exp()	24
5.1.2.36 exponential()	24
5.1.2.37 flatten()	25
5.1.2.38 gelu()	25
5.1.2.39 gemm() [1/2]	25
5.1.2.40 gemm() [2/2]	25
5.1.2.41 gemm_cpu()	25
5.1.2.42 get_default_session()	26
5.1.2.43 glorot_uniform()	26
5.1.2.44 hadamard_product() [1/2]	26
5.1.2.45 hadamard_product() [2/2]	26
5.1.2.46 hard_sigmoid()	26
5.1.2.47 has_inf()	26
5.1.2.48 has_nan()	27
5.1.2.49 hinge_loss()	27
5.1.2.50 identity()	27
5.1.2.51 img2col()	27
5.1.2.52 Input()	27
5.1.2.53 instance_normalization()	27
5.1.2.54 is_valid()	28
5.1.2.55 leaky_relu()	28
5.1.2.56 LeakyReLU()	28
5.1.2.57 linspace()	28
5.1.2.58 load_tensor()	28
5.1.2.59 log()	28
5.1.2.60 mae()	29
5.1.2.61 make_compiled_model()	29
5.1.2.62 max() [1/2]	29
5.1.2.63 max() [2/2]	29

5.1.2.64 max_pooling_2d()	29
5.1.2.65 maximum()	29
5.1.2.66 mean() [1/2]	30
5.1.2.67 mean() [2/2]	30
5.1.2.68 mean_absolute_error()	30
5.1.2.69 mean_reduce()	30
5.1.2.70 mean_squared_error()	30
5.1.2.71 mean_squared_logarithmic_error()	30
5.1.2.72 min() [1/2]	31
5.1.2.73 min() [2/2]	31
5.1.2.74 minus() [1/2]	31
5.1.2.75 minus() [2/2]	31
5.1.2.76 mse()	31
5.1.2.77 Multiply()	32
5.1.2.78 multiply() [1/2]	32
5.1.2.79 multiply() [2/2]	32
5.1.2.80 negative()	32
5.1.2.81 negative_relu()	32
5.1.2.82 norm()	33
5.1.2.83 normalization_batch()	33
5.1.2.84 normalization_instance()	33
5.1.2.85 ones()	33
5.1.2.86 ones_like()	33
5.1.2.87 operator!=()	33
5.1.2.88 operator*() [1/4]	34
5.1.2.89 operator*() [2/4]	34
5.1.2.90 operator*() [3/4]	34
5.1.2.91 operator*() [4/4]	34
5.1.2.92 operator+() [1/4]	34
5.1.2.93 operator+() [2/4]	34
5.1.2.94 operator+() [3/4]	35
5.1.2.95 operator+() [4/4]	35
5.1.2.96 operator-() [1/4]	35
5.1.2.97 operator-() [2/4]	35
5.1.2.98 operator-() [3/4]	35
5.1.2.99 operator-() [4/4]	35
5.1.2.100 operator/()	36
5.1.2.101 operator<()	36
5.1.2.102 operator<<()	36
5.1.2.103 operator<=()	36
5.1.2.104 operator==(1/2)	36
5.1.2.105 operator==(2/2)	36

5.1.2.106 operator>()	37
5.1.2.107 operator>=()	37
5.1.2.108 plus()	37
5.1.2.109 randn()	37
5.1.2.110 randn_like()	37
5.1.2.111 random()	38
5.1.2.112 random_like()	38
5.1.2.113 random_normal_like()	38
5.1.2.114 read_tensor()	38
5.1.2.115 reduce()	39
5.1.2.116 reduce_mean() [1/2]	39
5.1.2.117 reduce_mean() [2/2]	39
5.1.2.118 reduce_sum() [1/2]	39
5.1.2.119 reduce_sum() [2/2]	39
5.1.2.120 relu()	39
5.1.2.121 ReLU()	40
5.1.2.122 repeat()	40
5.1.2.123 replace_placeholder_with_expression()	40
5.1.2.124 repmat()	40
5.1.2.125 Reshape()	41
5.1.2.126 reshape() [1/2]	41
5.1.2.127 reshape() [2/2]	41
5.1.2.128 save_tensor()	41
5.1.2.129 selu()	41
5.1.2.130 sigmoid()	41
5.1.2.131 Softmax()	42
5.1.2.132 softmax() [1/2]	42
5.1.2.133 softmax() [2/2]	42
5.1.2.134 softplus()	42
5.1.2.135 softsign()	42
5.1.2.136 square()	42
5.1.2.137 squared_loss()	43
5.1.2.138 squeeze()	43
5.1.2.139 standard_deviation()	43
5.1.2.140 Subtract()	43
5.1.2.141 sum() [1/2]	43
5.1.2.142 sum() [2/2]	44
5.1.2.143 sum_reduce()	44
5.1.2.144 tanh()	44
5.1.2.145 transpose()	44
5.1.2.146 truncated_normal()	44
5.1.2.147 up_sampling_2d()	44

5.1.2.148 update_cuda_gemm_threshold()	45
5.1.2.149 variance()	45
5.1.2.150 write_tensor()	45
5.1.2.151 zeros()	45
5.1.2.152 zeros_like()	45
5.1.3 Variable Documentation	45
5.1.3.1 __version__	45
5.1.3.2 Adadelta	46
5.1.3.3 Adagrad	46
5.1.3.4 Adam	46
5.1.3.5 Binary_Operator	46
5.1.3.6 blas_mode	46
5.1.3.7 CategoricalCrossentropy	47
5.1.3.8 Constant	47
5.1.3.9 cuda_gemm_threshold	47
5.1.3.10 cuda_mode	47
5.1.3.11 debug_mode	47
5.1.3.12 eps	47
5.1.3.13 Expression	48
5.1.3.14 Hinge	48
5.1.3.15 is_binary_operator_v	48
5.1.3.16 is_constant_v	48
5.1.3.17 is_place_holder_v	48
5.1.3.18 is_tensor_v	48
5.1.3.19 is_unary_operator_v	49
5.1.3.20 is_value_v	49
5.1.3.21 is_variable_v	49
5.1.3.22 is_windows_platform	49
5.1.3.23 learning_phase	49
5.1.3.24 lstm	49
5.1.3.25 make_binary_operator	50
5.1.3.26 make_unary_operator	50
5.1.3.27 MeanAbsoluteError	50
5.1.3.28 MeanSquaredError	50
5.1.3.29 Operator	51
5.1.3.30 Place_Holder	51
5.1.3.31 random_generator	51
5.1.3.32 random_seed	51
5.1.3.33 RMSprop	51
5.1.3.34 SGD	51
5.1.3.35 Tensor	52
5.1.3.36 Unary_Operator	52

5.1.3.37 Value	52
5.1.3.38 Variable	52
5.1.3.39 version	52
5.1.3.40 visible_device	52
5.2 ceras::dataset Namespace Reference	52
5.3 ceras::dataset::mnist Namespace Reference	53
5.3.1 Function Documentation	53
5.3.1.1 load_data()	53
6 Class Documentation	55
6.1 ceras::adadelta< Loss, T > Struct Template Reference	55
6.1.1 Member Typedef Documentation	55
6.1.1.1 tensor_type	56
6.1.2 Constructor & Destructor Documentation	56
6.1.2.1 adadelta()	56
6.1.3 Member Function Documentation	56
6.1.3.1 forward()	56
6.1.4 Member Data Documentation	56
6.1.4.1 iterations_	56
6.1.4.2 learning_rate_	56
6.1.4.3 loss_	57
6.1.4.4 rho_	57
6.2 ceras::adagrad< Loss, T > Struct Template Reference	57
6.2.1 Member Typedef Documentation	57
6.2.1.1 tensor_type	58
6.2.2 Constructor & Destructor Documentation	58
6.2.2.1 adagrad()	58
6.2.3 Member Function Documentation	58
6.2.3.1 forward()	58
6.2.4 Member Data Documentation	58
6.2.4.1 decay_	58
6.2.4.2 iterations_	58
6.2.4.3 learning_rate_	59
6.2.4.4 loss_	59
6.3 ceras::adam< Loss, T > Struct Template Reference	59
6.3.1 Member Typedef Documentation	60
6.3.1.1 tensor_type	60
6.3.2 Constructor & Destructor Documentation	60
6.3.2.1 adam()	60
6.3.3 Member Function Documentation	60
6.3.3.1 forward()	60
6.3.4 Member Data Documentation	60

6.3.4.1 amsgrad_	60
6.3.4.2 beta_1_	61
6.3.4.3 beta_2_	61
6.3.4.4 iterations_	61
6.3.4.5 learning_rate_	61
6.3.4.6 loss_	61
6.4 ceras::binary_operator< Lhs_Operator, Rh_Operator, Forward_Action, Backward_Action > Struct Template Reference	61
6.4.1 Member Typedef Documentation	62
6.4.1.1 tensor_type	62
6.4.1.2 wrapped_lhs_operator_type	62
6.4.1.3 wrapped_rhs_operator_type	63
6.4.2 Constructor & Destructor Documentation	63
6.4.2.1 binary_operator()	63
6.4.3 Member Function Documentation	63
6.4.3.1 backward()	63
6.4.3.2 forward()	63
6.4.3.3 reset_states()	64
6.4.4 Member Data Documentation	64
6.4.4.1 backward_action_	64
6.4.4.2 forward_action_	64
6.4.4.3 lhs_input_data_	64
6.4.4.4 lhs_op_	64
6.4.4.5 output_data_	65
6.4.4.6 reset_action_	65
6.4.4.7 rhs_input_data_	65
6.4.4.8 rhs_op_	65
6.5 ceras::compiled_model< Model, Optimizer, Loss > Struct Template Reference	65
6.5.1 Member Typedef Documentation	66
6.5.1.1 io_layer_type	66
6.5.2 Constructor & Destructor Documentation	66
6.5.2.1 compiled_model()	66
6.5.3 Member Function Documentation	67
6.5.3.1 evaluate()	67
6.5.3.2 fit()	67
6.5.3.3 operator>()	68
6.5.3.4 predict()	68
6.5.3.5 train_on_batch()	68
6.5.4 Member Data Documentation	69
6.5.4.1 compiled_optimizer_	69
6.5.4.2 ground_truth_place_holder_	69
6.5.4.3 input_place_holder_	69

6.5.4.4 loss_	69
6.5.4.5 model_	69
6.5.4.6 optimizer_	70
6.5.4.7 optimizer_type	70
6.6 ceras::constant< Tsor > Struct Template Reference	70
6.6.1 Constructor & Destructor Documentation	70
6.6.1.1 constant()	70
6.6.2 Member Function Documentation	70
6.6.2.1 backward()	71
6.6.2.2 forward()	71
6.6.2.3 shape()	71
6.6.3 Member Data Documentation	71
6.6.3.1 data_	71
6.7 ceras::gradient_descent< Loss, T > Struct Template Reference	71
6.7.1 Member Typedef Documentation	72
6.7.1.1 tensor_type	72
6.7.2 Constructor & Destructor Documentation	72
6.7.2.1 gradient_descent()	72
6.7.3 Member Function Documentation	72
6.7.3.1 forward()	72
6.7.4 Member Data Documentation	72
6.7.4.1 learning_rate_	73
6.7.4.2 loss_	73
6.7.4.3 momentum_	73
6.8 ceras::is_binary_operator< T > Struct Template Reference	73
6.9 ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > > Struct Template Reference	74
6.10 ceras::is_constant< T > Struct Template Reference	74
6.11 ceras::is_constant< constant< Tsor > > Struct Template Reference	74
6.12 ceras::is_place_holder< T > Struct Template Reference	75
6.13 ceras::is_place_holder< place_holder< Tsor > > Struct Template Reference	75
6.14 ceras::is_tensor< T > Struct Template Reference	75
6.15 ceras::is_tensor< tensor< T, A > > Struct Template Reference	76
6.16 ceras::is_unary_operator< T > Struct Template Reference	76
6.17 ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > > Struct Template Reference	76
6.18 ceras::is_value< T > Struct Template Reference	77
6.19 ceras::is_value< value< T > > Struct Template Reference	77
6.20 ceras::is_variable< T > Struct Template Reference	77
6.21 ceras::is_variable< variable< Tsor > > Struct Template Reference	78
6.22 ceras::model< Ex, Ph > Struct Template Reference	78
6.22.1 Detailed Description	78
6.22.2 Member Typedef Documentation	79

6.22.2.1 input_layer_type	79
6.22.2.2 output_layer_type	79
6.22.3 Constructor & Destructor Documentation	79
6.22.3.1 model()	79
6.22.4 Member Function Documentation	79
6.22.4.1 compile()	80
6.22.4.2 input()	80
6.22.4.3 operator>()	80
6.22.4.4 output()	81
6.22.4.5 predict()	81
6.22.5 Member Data Documentation	81
6.22.5.1 expression_	81
6.22.5.2 place_holder_	82
6.23 ceras::place_holder< T> Struct Template Reference	82
6.23.1 Member Typedef Documentation	82
6.23.1.1 tensor_type	82
6.23.2 Constructor & Destructor Documentation	83
6.23.2.1 place_holder() [1/4]	83
6.23.2.2 place_holder() [2/4]	83
6.23.2.3 place_holder() [3/4]	83
6.23.2.4 place_holder() [4/4]	83
6.23.3 Member Function Documentation	83
6.23.3.1 backward()	83
6.23.3.2 bind()	84
6.23.3.3 forward()	84
6.23.3.4 operator=() [1/2]	84
6.23.3.5 operator=() [2/2]	84
6.23.3.6 reset()	84
6.24 ceras::place_holder_state< T> Struct Template Reference	84
6.24.1 Member Data Documentation	85
6.24.1.1 data_	85
6.24.1.2 shape_hint_	85
6.25 ceras::rmsprop< Loss, T> Struct Template Reference	85
6.25.1 Member Typedef Documentation	86
6.25.1.1 tensor_type	86
6.25.2 Constructor & Destructor Documentation	86
6.25.2.1 rmsprop()	86
6.25.3 Member Function Documentation	86
6.25.3.1 forward()	86
6.25.4 Member Data Documentation	86
6.25.4.1 decay_	87
6.25.4.2 iterations_	87

6.25.4.3 learning_rate_	87
6.25.4.4 loss_	87
6.25.4.5 rho_	87
6.26 ceras::session< Tsor > Struct Template Reference	87
6.26.1 Member Typedef Documentation	88
6.26.1.1 place_holder_type	88
6.26.1.2 variable_state_type	88
6.26.1.3 variable_type	88
6.26.2 Constructor & Destructor Documentation	89
6.26.2.1 session() [1/3]	89
6.26.2.2 session() [2/3]	89
6.26.2.3 session() [3/3]	89
6.26.2.4 ~session()	89
6.26.3 Member Function Documentation	89
6.26.3.1 bind()	89
6.26.3.2 deserialize()	90
6.26.3.3 operator=() [1/2]	90
6.26.3.4 operator=() [2/2]	90
6.26.3.5 rebind()	90
6.26.3.6 remember()	90
6.26.3.7 restore()	90
6.26.3.8 run()	91
6.26.3.9 save()	91
6.26.3.10 serialize()	91
6.26.3.11 tap()	91
6.26.4 Member Data Documentation	91
6.26.4.1 place_holders_	91
6.26.4.2 variables_	92
6.27 ceras::sgd< Loss, T > Struct Template Reference	92
6.27.1 Member Typedef Documentation	92
6.27.1.1 tensor_type	93
6.27.2 Constructor & Destructor Documentation	93
6.27.2.1 sgd()	93
6.27.3 Member Function Documentation	93
6.27.3.1 forward()	93
6.27.4 Member Data Documentation	93
6.27.4.1 decay_	93
6.27.4.2 iterations_	93
6.27.4.3 learning_rate_	94
6.27.4.4 loss_	94
6.27.4.5 momentum_	94
6.27.4.6 nesterov_	94

6.28 ceras::tensor< T, Allocator > Struct Template Reference	94
6.28.1 Member Typedef Documentation	96
6.28.1.1 allocator	96
6.28.1.2 self_type	96
6.28.1.3 shared_vector	96
6.28.1.4 value_type	96
6.28.1.5 vector_type	96
6.28.2 Constructor & Destructor Documentation	96
6.28.2.1 tensor() [1/7]	96
6.28.2.2 tensor() [2/7]	97
6.28.2.3 tensor() [3/7]	97
6.28.2.4 tensor() [4/7]	97
6.28.2.5 tensor() [5/7]	97
6.28.2.6 tensor() [6/7]	97
6.28.2.7 tensor() [7/7]	97
6.28.3 Member Function Documentation	98
6.28.3.1 as_scalar()	98
6.28.3.2 as_type()	98
6.28.3.3 begin() [1/2]	98
6.28.3.4 begin() [2/2]	98
6.28.3.5 cbegin()	98
6.28.3.6 cend()	98
6.28.3.7 copy()	99
6.28.3.8 creep_to()	99
6.28.3.9 data() [1/2]	99
6.28.3.10 data() [2/2]	99
6.28.3.11 deep_copy() [1/2]	99
6.28.3.12 deep_copy() [2/2]	99
6.28.3.13 empty()	100
6.28.3.14 end() [1/2]	100
6.28.3.15 end() [2/2]	100
6.28.3.16 map()	100
6.28.3.17 ndim()	100
6.28.3.18 operator*=() [1/2]	100
6.28.3.19 operator*=() [2/2]	101
6.28.3.20 operator+=() [1/2]	101
6.28.3.21 operator+=() [2/2]	101
6.28.3.22 operator-()	101
6.28.3.23 operator==() [1/2]	101
6.28.3.24 operator==() [2/2]	101
6.28.3.25 operator/=() [1/2]	102
6.28.3.26 operator/=() [2/2]	102

6.28.3.27 operator=() [1/2]	102
6.28.3.28 operator=() [2/2]	102
6.28.3.29 operator[]() [1/2]	102
6.28.3.30 operator[]() [2/2]	102
6.28.3.31 reset()	103
6.28.3.32 reshape()	103
6.28.3.33 resize()	103
6.28.3.34 shape()	103
6.28.3.35 shrink_to()	103
6.28.3.36 size()	103
6.28.3.37 slice()	104
6.28.4 Member Data Documentation	104
6.28.4.1 memory_offset_	104
6.28.4.2 shape_	104
6.28.4.3 vector_	104
6.29 ceras::tensor_deduction< L, R > Struct Template Reference	104
6.29.1 Member Typedef Documentation	105
6.29.1.1 op_type	105
6.29.1.2 tensor_type	105
6.30 ceras::unary_operator< Operator, Forward_Action, Backward_Action > Struct Template Reference	105
6.30.1 Member Typedef Documentation	106
6.30.1.1 wrapped_operator_type	106
6.30.2 Constructor & Destructor Documentation	106
6.30.2.1 unary_operator()	106
6.30.3 Member Function Documentation	106
6.30.3.1 backward()	106
6.30.3.2 forward()	107
6.30.3.3 reset_states()	107
6.30.4 Member Data Documentation	107
6.30.4.1 backward_action_	107
6.30.4.2 forward_action_	107
6.30.4.3 input_data_	107
6.30.4.4 op_	107
6.30.4.5 output_data_	108
6.30.4.6 reset_action_	108
6.30.4.7 tensor_type	108
6.31 ceras::value< T > Struct Template Reference	108
6.31.1 Member Typedef Documentation	109
6.31.1.1 value_type	109
6.31.2 Constructor & Destructor Documentation	109
6.31.2.1 value() [1/4]	109
6.31.2.2 value() [2/4]	109

6.31.2.3 value() [3/4]	109
6.31.2.4 value() [4/4]	109
6.31.3 Member Function Documentation	110
6.31.3.1 backward()	110
6.31.3.2 forward()	110
6.31.3.3 operator=() [1/2]	110
6.31.3.4 operator=() [2/2]	110
6.31.4 Member Data Documentation	110
6.31.4.1 data_	110
6.32 ceras::variable< Tsor > Struct Template Reference	111
6.32.1 Member Typedef Documentation	111
6.32.1.1 tensor_type	112
6.32.2 Constructor & Destructor Documentation	112
6.32.2.1 variable() [1/4]	112
6.32.2.2 variable() [2/4]	112
6.32.2.3 variable() [3/4]	112
6.32.2.4 variable() [4/4]	112
6.32.3 Member Function Documentation	112
6.32.3.1 backward()	113
6.32.3.2 contexts() [1/2]	113
6.32.3.3 contexts() [2/2]	113
6.32.3.4 data() [1/2]	113
6.32.3.5 data() [2/2]	113
6.32.3.6 forward()	113
6.32.3.7 gradient() [1/2]	113
6.32.3.8 gradient() [2/2]	114
6.32.3.9 operator=() [1/2]	114
6.32.3.10 operator=() [2/2]	114
6.32.3.11 reset()	114
6.32.3.12 reset_states()	114
6.32.3.13 shape()	114
6.32.4 Member Data Documentation	114
6.32.4.1 state_	115
6.32.4.2 stateful_	115
6.32.4.3 trainable_	115
6.33 ceras::variable_state< Tsor > Struct Template Reference	115
6.33.1 Member Data Documentation	115
6.33.1.1 contexts_	115
6.33.1.2 data_	116
6.33.1.3 gradient_	116
6.34 ceras::view_2d< T > Struct Template Reference	116
6.34.1 Constructor & Destructor Documentation	116

6.34.1.1 view_2d() [1/3]	117
6.34.1.2 view_2d() [2/3]	117
6.34.1.3 view_2d() [3/3]	117
6.34.2 Member Function Documentation	117
6.34.2.1 data() [1/2]	117
6.34.2.2 data() [2/2]	117
6.34.2.3 operator[]() [1/2]	118
6.34.2.4 operator[]() [2/2]	118
6.34.2.5 shape()	118
6.34.2.6 size()	118
6.34.3 Member Data Documentation	118
6.34.3.1 col_	118
6.34.3.2 data_	118
6.34.3.3 row_	119
6.34.3.4 transposed_	119
6.35 ceras::view_3d< T > Struct Template Reference	119
6.35.1 Constructor & Destructor Documentation	119
6.35.1.1 view_3d()	119
6.35.2 Member Function Documentation	120
6.35.2.1 operator[]() [1/2]	120
6.35.2.2 operator[]() [2/2]	120
6.35.3 Member Data Documentation	120
6.35.3.1 channel_	120
6.35.3.2 col_	120
6.35.3.3 data_	120
6.35.3.4 row_	121
6.36 ceras::view_4d< T > Struct Template Reference	121
6.36.1 Detailed Description	121
6.36.2 Constructor & Destructor Documentation	121
6.36.2.1 view_4d()	122
6.36.3 Member Function Documentation	123
6.36.3.1 operator[]() [1/2]	123
6.36.3.2 operator[]() [2/2]	123
6.36.4 Member Data Documentation	124
6.36.4.1 batch_size_	124
6.36.4.2 channel_	124
6.36.4.3 col_	124
6.36.4.4 data_	124
6.36.4.5 row_	124
7 File Documentation	125
7.1 /data/structured_folders/workspace/github.repo/ceras/include/activation.hpp File Reference	125

7.2 /data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp File Reference	126
7.3 /data/structured_folders/workspace/github.repo/ceras/include/config.hpp File Reference	126
7.4 /data/structured_folders/workspace/github.repo/ceras/include/constant.hpp File Reference	127
7.5 /data/structured_folders/workspace/github.repo/ceras/include/dataset.hpp File Reference	127
7.6 /data/structured_folders/workspace/github.repo/ceras/include/includes.hpp File Reference	128
7.6.1 Macro Definition Documentation	128
7.6.1.1 STB_IMAGE_IMPLEMENTATION	129
7.6.1.2 STB_IMAGE_RESIZE_IMPLEMENTATION	129
7.6.1.3 STB_IMAGE_WRITE_IMPLEMENTATION	129
7.7 /data/structured_folders/workspace/github.repo/ceras/include/keras.hpp File Reference	129
7.8 /data/structured_folders/workspace/github.repo/ceras/include/layer.hpp File Reference	129
7.9 /data/structured_folders/workspace/github.repo/ceras/include/loss.hpp File Reference	130
7.10 /data/structured_folders/workspace/github.repo/ceras/include/model.hpp File Reference	131
7.11 /data/structured_folders/workspace/github.repo/ceras/include/operation.hpp File Reference	132
7.12 /data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp File Reference	134
7.13 /data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp File Reference	135
7.14 /data/structured_folders/workspace/github.repo/ceras/include/recurrent_operation.hpp File Reference	136
7.15 /data/structured_folders/workspace/github.repo/ceras/include/session.hpp File Reference	137
7.16 /data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp File Reference	137
7.17 /data/structured_folders/workspace/github.repo/ceras/include/value.hpp File Reference	141
7.18 /data/structured_folders/workspace/github.repo/ceras/include/variable.hpp File Reference	141
Index	143

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

ceras	9
ceras::dataset	52
ceras::dataset::mnist	53

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

ceras::compiled_model< Model, Optimizer, Loss >	65
ceras::constant< Tzor >	70
enable_id	
ceras::adadelta< Loss, T >	55
ceras::adagrad< Loss, T >	57
ceras::adam< Loss, T >	59
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >	61
ceras::gradient_descent< Loss, T >	71
ceras::place_holder< Tzor >	82
ceras::rmsprop< Loss, T >	85
ceras::sgd< Loss, T >	92
ceras::tensor< T, Allocator >	94
ceras::unary_operator< Operator, Forward_Action, Backward_Action >	105
ceras::variable< Tzor >	111
enable_shared	
ceras::adadelta< Loss, T >	55
ceras::adagrad< Loss, T >	57
ceras::adam< Loss, T >	59
ceras::gradient_descent< Loss, T >	71
ceras::rmsprop< Loss, T >	85
ceras::sgd< Loss, T >	92
enable_shared_state	
ceras::place_holder< Tzor >	82
std::false_type	
ceras::is_binary_operator< T >	73
ceras::is_constant< T >	74
ceras::is_place_holder< T >	75
ceras::is_tensor< T >	75
ceras::is_unary_operator< T >	76
ceras::is_value< T >	77
ceras::is_variable< T >	77
ceras::model< Ex, Ph >	78
ceras::place_holder_state< Tzor >	84
ceras::session< Tzor >	87
ceras::tensor_deduction< L, R >	104

ceras::tensor_deduction< Lhs_Operator, Rhs_Operator >	104
std::true_type	
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > >	74
ceras::is_constant< constant< Tsor > >	74
ceras::is_place_holder< place_holder< Tsor > >	75
ceras::is_tensor< tensor< T, A > >	76
ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >	76
ceras::is_value< value< T > >	77
ceras::is_variable< variable< Tsor > >	78
ceras::value< T >	108
ceras::variable_state< Tsor >	115
ceras::view_2d< T >	116
ceras::view_3d< T >	119
ceras::view_4d< T >	121

Chapter 3

Class Index

3.1 Class List

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

ceras::adadelta< Loss, T >	55
ceras::adagrad< Loss, T >	57
ceras::adam< Loss, T >	59
ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >	61
ceras::compiled_model< Model, Optimizer, Loss >	65
ceras::constant< Tsor >	70
ceras::gradient_descent< Loss, T >	71
ceras::is_binary_operator< T >	73
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > >	74
ceras::is_constant< T >	74
ceras::is_constant< constant< Tsor > >	74
ceras::is_place_holder< T >	75
ceras::is_place_holder< place_holder< Tsor > >	75
ceras::is_tensor< T >	75
ceras::is_tensor< tensor< T, A > >	76
ceras::is_unary_operator< T >	76
ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >	76
ceras::is_value< T >	77
ceras::is_value< value< T > >	77
ceras::is_variable< T >	77
ceras::is_variable< variable< Tsor > >	78
ceras::model< Ex, Ph >	78
ceras::place_holder< Tsor >	82
ceras::place_holder_state< Tsor >	84
ceras::rmsprop< Loss, T >	85
ceras::session< Tsor >	87
ceras::sgd< Loss, T >	92
ceras::tensor< T, Allocator >	94
ceras::tensor_deduction< L, R >	104
ceras::unary_operator< Operator, Forward_Action, Backward_Action >	105
ceras::value< T >	108
ceras::variable< Tsor >	111
ceras::variable_state< Tsor >	115
ceras::view_2d< T >	116
ceras::view_3d< T >	119
ceras::view_4d< T >	121

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	125
/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp	126
/data/structured_folders/workspace/github.repo/ceras/include/config.hpp	126
/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp	127
/data/structured_folders/workspace/github.repo/ceras/include/dataset.hpp	127
/data/structured_folders/workspace/github.repo/ceras/include/includes.hpp	128
/data/structured_folders/workspace/github.repo/ceras/include/keras.hpp	129
/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp	129
/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp	130
/data/structured_folders/workspace/github.repo/ceras/include/model.hpp	131
/data/structured_folders/workspace/github.repo/ceras/include/operation.hpp	132
/data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp	134
/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp	135
/data/structured_folders/workspace/github.repo/ceras/include/recurrent_operation.hpp	136
/data/structured_folders/workspace/github.repo/ceras/include/session.hpp	137
/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp	137
/data/structured_folders/workspace/github.repo/ceras/include/value.hpp	141
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp	141

Chapter 5

Namespace Documentation

5.1 ceras Namespace Reference

Namespaces

- [dataset](#)

Classes

- struct [constant](#)
- struct [is_constant](#)
- struct [is_constant< constant< T_{src} > >](#)
- 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, Rh_s_Operator, Forward_Action, Backward_Action > >](#)
- struct [sgd](#)
- struct [adagrad](#)
- struct [rmsprop](#)
- struct [adadel_ta](#)
- struct [adam](#)
- struct [gradient_descent](#)
- struct [place_holder_state](#)
- struct [place_holder](#)
- struct [is_place_holder](#)
- struct [is_place_holder< place_holder< T_{src} > >](#)
- struct [session](#)
- 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 [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](#) = [adadelat](#)< 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})
- auto [Dense](#) (unsigned long output_size, unsigned long input_size)
- auto [BatchNormalization](#) (std::vector< unsigned long > const &shape, float threshold=0.95f)
- 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
- 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 [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, 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 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 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`
- `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<typename T = double>`
`requires std::floating_point< T > auto normalization_instance (T const momentum=0.98) noexcept`
- `template<typename T >`
`requires std::floating_point< T > auto instance_normalization (T const momentum=0.98) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto concatenate (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `auto concatenate (unsigned long axe=-1)`

- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto concat (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `auto concat (unsigned long axe=-1)`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto maximum (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<typename T = float>`
`requires std::floating_point< T > auto random_normal_like (T mean=0.0, T stddev=1.0) noexcept`
- `template<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<Expression Lhs_Expression, Variable Rhs_Variable>`
`constexpr auto copy (Lhs_Expression const &lhs_ex, Rhs_Variable const &rhs_va) noexcept`
- `template<Tensor Tsor>`
`std::reference_wrapper< 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 > gloriot_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>`
`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

- constexpr unsigned long `version` = 20210418UL
- constexpr unsigned long `__version__` = `version`
- constexpr unsigned long `is_windows_platform` = 0
- constexpr unsigned long `debug_mode` = 1
- constexpr unsigned long `blas_mode` = 0
- constexpr unsigned long `cuda_mode` = 0
- int `visible_device` = 0
- unsigned long `cuda_gemm_threshold` = 0UL
- constexpr double `eps` = 1.0e-8
- int `learning_phase` = 1
- `template<class T >`
constexpr bool `is_constant_v` = `is_constant<T>::value`
- `template<typename T >`
concept `Constant` = `is_constant_v<T>`
- auto `MeanSquaredError`
- auto `MeanAbsoluteError`
- auto `Hinge`
- auto `CategoricalCrossentropy`
- 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>`
- auto `lstm`
- 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 batch_normalization()

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

5.1.2.11 BatchNormalization()

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

5.1.2.12 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]
```

5.1.2.13 clip() [2/2]

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

5.1.2.14 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]
```

5.1.2.15 concat() [2/2]

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

5.1.2.16 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]
```

5.1.2.17 concatenate() [2/3]

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

5.1.2.18 concatenate() [3/3]

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

5.1.2.19 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)
```

5.1.2.20 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} ) [inline]
```

5.1.2.21 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]
```

5.1.2.22 copy() [1/2]

```
template<Expression Lhs_Expression, Variable Rhs_Variable>
constexpr auto ceras::copy (
    Lhs_Expression const & lhs_ex,
    Rhs_Variable const & rhs_va ) [constexpr], [noexcept]
```

5.1.2.23 copy() [2/2]

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

5.1.2.24 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]
```


5.1.2.25 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]
```

5.1.2.26 deep_copy()

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

5.1.2.27 Dense()

```
auto ceras::Dense (
    unsigned long output_size,
    unsigned long input_size ) [inline]
```

5.1.2.28 drop_out()

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

5.1.2.29 elementwise_divide()

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

5.1.2.30 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]
```

5.1.2.31 elementwise_product() [2/2]

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

5.1.2.32 elu()

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

5.1.2.33 ELU()

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

Exponential Linear Unit.

5.1.2.34 empty()

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

5.1.2.35 exp()

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

5.1.2.36 exponential()

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

5.1.2.37 flatten()

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

5.1.2.38 gelu()

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

5.1.2.39 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 )
```

5.1.2.40 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 )
```

5.1.2.41 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 )
```

5.1.2.42 get_default_session()

```
template<Tensor Tsor>
std::reference_wrapper< session< Tsor > > ceras::get_default_session ( )
```

5.1.2.43 glorot_uniform()

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

5.1.2.44 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]
```

5.1.2.45 hadamard_product() [2/2]

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

5.1.2.46 hard_sigmoid()

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

5.1.2.47 has_inf()

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

5.1.2.48 has_nan()

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

5.1.2.49 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]
```

5.1.2.50 identity()

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

5.1.2.51 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]
```

5.1.2.52 Input()

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

5.1.2.53 instance_normalization()

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

5.1.2.54 is_valid()

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

5.1.2.55 leaky_relu()

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

5.1.2.56 LeakyReLU()

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

leaky relu activation function.

5.1.2.57 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]
```

5.1.2.58 load_tensor()

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

5.1.2.59 log()

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

5.1.2.60 mae()

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

5.1.2.61 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]
```

5.1.2.62 max() [1/2]

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

5.1.2.63 max() [2/2]

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

5.1.2.64 max_pooling_2d()

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

5.1.2.65 maximum()

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

5.1.2.66 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]
```

5.1.2.67 mean() [2/2]

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

5.1.2.68 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]
```

5.1.2.69 mean_reduce()

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

5.1.2.70 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]
```

5.1.2.71 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]
```


5.1.2.72 min() [1/2]

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

5.1.2.73 min() [2/2]

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

5.1.2.74 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]
```

5.1.2.75 minus() [2/2]

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

5.1.2.76 mse()

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

5.1.2.77 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 };
```

5.1.2.78 multiply() [1/2]

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

5.1.2.79 multiply() [2/2]

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

5.1.2.80 negative()

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

5.1.2.81 negative_relu()

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

5.1.2.82 norm()

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

5.1.2.83 normalization_batch()

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

5.1.2.84 normalization_instance()

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

5.1.2.85 ones()

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

5.1.2.86 ones_like()

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

5.1.2.87 operator"!="()

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

5.1.2.88 operator*() [1/4]

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

5.1.2.89 operator*() [2/4]

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

5.1.2.90 operator*() [3/4]

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

5.1.2.91 operator*() [4/4]

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

5.1.2.92 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]
```

5.1.2.93 operator+() [2/4]

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

5.1.2.94 operator+() [3/4]

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

5.1.2.95 operator+() [4/4]

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

5.1.2.96 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]
```

5.1.2.97 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/()

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

5.1.2.101 operator<()

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

5.1.2.102 operator<<()

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

5.1.2.103 operator<=()

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

5.1.2.104 operator==() [1/2]

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

5.1.2.105 operator==() [2/2]

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

5.1.2.106 operator>()

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

5.1.2.107 operator>=()

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

5.1.2.108 plus()

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

5.1.2.109 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} )
```

5.1.2.110 randn_like()

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

5.1.2.111 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} )
```

5.1.2.112 random_like()

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

5.1.2.113 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)
```

5.1.2.114 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 )
```


5.1.2.115 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]
```

5.1.2.116 reduce_mean() [1/2]

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

5.1.2.117 reduce_mean() [2/2]

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

5.1.2.118 reduce_sum() [1/2]

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

5.1.2.119 reduce_sum() [2/2]

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

5.1.2.120 relu()

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

5.1.2.121 ReLU()

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

Rectified Linear Unit activation function.

5.1.2.122 repeat()

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

5.1.2.123 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*

5.1.2.124 repmat()

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

5.1.2.125 Reshape()

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

Reshapes inputs into the given shape.

5.1.2.126 reshape() [1/2]

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

5.1.2.127 reshape() [2/2]

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

5.1.2.128 save_tensor()

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

5.1.2.129 selu()

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

5.1.2.130 sigmoid()

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

5.1.2.131 Softmax()

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

Softmax activation function.

5.1.2.132 softmax() [1/2]

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

5.1.2.133 softmax() [2/2]

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

5.1.2.134 softplus()

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

5.1.2.135 softsign()

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

5.1.2.136 square()

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

5.1.2.137 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]
```

5.1.2.138 squeeze()

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

5.1.2.139 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]
```

5.1.2.140 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 };
```

5.1.2.141 sum() [1/2]

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

5.1.2.142 sum() [2/2]

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

5.1.2.143 sum_reduce()

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

5.1.2.144 tanh()

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

5.1.2.145 transpose()

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

5.1.2.146 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} )
```

5.1.2.147 up_sampling_2d()

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

5.1.2.148 update_cuda_gemm_threshold()

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

5.1.2.149 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]
```

5.1.2.150 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 )
```

5.1.2.151 zeros()

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

5.1.2.152 zeros_like()

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

5.1.3 Variable Documentation**5.1.3.1 __version__**

```
constexpr unsigned long ceras::__version__ = version [inline], [constexpr]
```

5.1.3.2 Adadelta

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

Initial value:

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

5.1.3.3 Adagrad

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

Initial value:

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

5.1.3.4 Adam

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

Initial value:

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

5.1.3.5 Binary_Operator

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

A type that represents a binary operator.

```
@concept Binary_Operator<>
```

5.1.3.6 blas_mode

```
constexpr unsigned long ceras::blas_mode = 0 [inline], [constexpr]
```


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 Constant

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

5.1.3.9 cuda_gemm_threshold

```
unsigned long ceras::cuda_gemm_threshold = 0UL [inline]
```

5.1.3.10 cuda_mode

```
constexpr unsigned long ceras::cuda_mode = 0 [inline], [constexpr]
```

5.1.3.11 debug_mode

```
constexpr unsigned long ceras::debug_mode = 1 [inline], [constexpr]
```

5.1.3.12 eps

```
constexpr double ceras::eps = 1.0e-8 [inline], [constexpr]
```

5.1.3.13 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<>

5.1.3.14 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 );
        };
    };
}
```

5.1.3.15 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.

5.1.3.16 is_constant_v

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

5.1.3.17 is_place_holder_v

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

5.1.3.18 is_tensor_v

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

5.1.3.19 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`.

5.1.3.20 is_value_v

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

5.1.3.21 is_variable_v

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

5.1.3.22 is_windows_platform

```
constexpr unsigned long ceras::is_windows_platform = 0 [inline], [constexpr]
```

Parameters

<i>is_windows_platform</i>	A constexpr helping ceras to select different behaviours. 1 for windows platform and 0 for non-windows platform.
----------------------------	--

Example usage:

```
if constexpr( is_windows_platform )
{
    call_windows_method();
}
else
{
    call_linux_method();
}
```

5.1.3.23 learning_phase

```
int ceras::learning_phase = 1 [inline]
```

5.1.3.24 lstm

```
auto ceras::lstm [inline]
```

5.1.3.25 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", std::function<void()> const& reset_action=[](){} ) noexcept
{
    return [&binary_forward_action, &binary_backward_action, &name, &reset_action] ( auto const& lhs_op,
    auto const& rhs_op ) noexcept
    {
        auto ans = binary_operator{ lhs_op, rhs_op, binary_forward_action, binary_backward_action,
        reset_action };
        ans.name_ = name;
        return ans;
    };
}
```

5.1.3.26 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", std::function<void()> reset_action = [](){} ) noexcept
{
    return [&unary_forward_action, &unary_backward_action, &name, &reset_action] ( auto const& op )
    noexcept
    {
        auto ans = unary_operator{ op, unary_forward_action, unary_backward_action, reset_action };
        ans.name_ = name;
        return ans;
    };
}
```

5.1.3.27 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 );
        };
    };
}
```

5.1.3.28 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 );
        };
    };
}
```

5.1.3.29 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<>
```

5.1.3.30 Place_Holder

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

5.1.3.31 random_generator

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

5.1.3.32 random_seed

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

5.1.3.33 RMSprop

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

Initial value:

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

5.1.3.34 SGD

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

Initial value:

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

5.1.3.35 Tensor

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

5.1.3.36 Unary_Operator

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

A type that represents an unary operator.

```
@concept Unary_Operator<>
```

5.1.3.37 Value

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

5.1.3.38 Variable

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

5.1.3.39 version

```
constexpr unsigned long ceras::version = 20210418UL [inline], [constexpr]
```

5.1.3.40 visible_device

```
int ceras::visible_device = 0 [inline]
```

5.2 ceras::dataset Namespace Reference

Namespaces

- [mnist](#)

5.3 ceras::dataset::mnist Namespace Reference

Functions

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

5.3.1 Function Documentation

5.3.1.1 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 "/dataset/mnist", should be updated if running the program somewhere else.
-------------	--

Returns

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

Example usage:

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

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

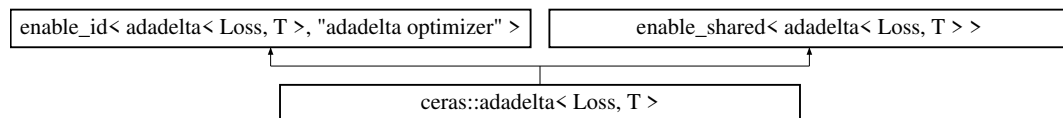
Chapter 6

Class Documentation

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

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

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



Public Types

- typedef [tensor](#)< T > [tensor_type](#)

Public Member Functions

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

Public Attributes

- Loss & [loss_](#)
- T [rho_](#)
- T [learning_rate_](#)
- unsigned long [iterations_](#)

6.1.1 Member Typedef Documentation

6.1.1.1 tensor_type

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

6.1.2 Constructor & Destructor Documentation

6.1.2.1 adadelta()

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

6.1.3 Member Function Documentation

6.1.3.1 forward()

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

6.1.4 Member Data Documentation

6.1.4.1 iterations_

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

6.1.4.2 learning_rate_

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

6.1.4.3 loss_

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

6.1.4.4 rho_

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

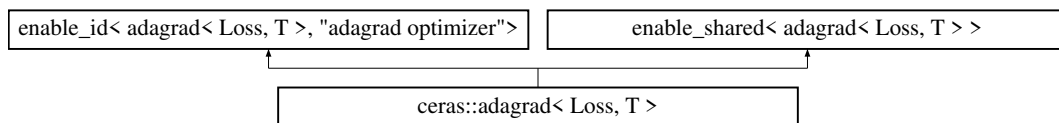
The documentation for this struct was generated from the following file:

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

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

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

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



Public Types

- typedef [tensor](#)< T > [tensor_type](#)

Public Member Functions

- [adagrad](#) (Loss &loss, std::size_t batch_size, T learning_rate=1.0e-1, T decay=0.0) noexcept
- void [forward](#) ()

Public Attributes

- Loss & [loss_](#)
- T [learning_rate_](#)
- T [decay_](#)
- unsigned long [iterations_](#)

6.2.1 Member Typedef Documentation

6.2.1.1 tensor_type

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

6.2.2 Constructor & Destructor Documentation

6.2.2.1 adagrad()

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

6.2.3 Member Function Documentation

6.2.3.1 forward()

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

6.2.4 Member Data Documentation

6.2.4.1 decay_

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

6.2.4.2 iterations_

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

6.2.4.3 learning_rate_

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

6.2.4.4 loss_

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

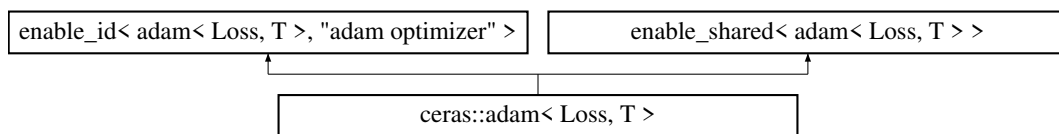
The documentation for this struct was generated from the following file:

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

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

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

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



Public Types

- typedef [tensor](#)< T > [tensor_type](#)

Public Member Functions

- [adam](#) (Loss &loss, std::size_t batch_size, T learning_rate=1.0e-1, T beta_1=0.9, T beta_2=0.999, bool amsgrad=false) noexcept
- void [forward](#) ()

Public Attributes

- Loss & [loss_](#)
- T [learning_rate_](#)
- T [beta_1_](#)
- T [beta_2_](#)
- bool [amsgrad_](#)
- unsigned long [iterations_](#)

6.3.1 Member Typedef Documentation

6.3.1.1 tensor_type

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

6.3.2 Constructor & Destructor Documentation

6.3.2.1 adam()

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

6.3.3 Member Function Documentation

6.3.3.1 forward()

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

6.3.4 Member Data Documentation

6.3.4.1 amsgrad_

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

6.3.4.2 beta_1_

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

6.3.4.3 beta_2_

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

6.3.4.4 iterations_

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

6.3.4.5 learning_rate_

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

6.3.4.6 loss_

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

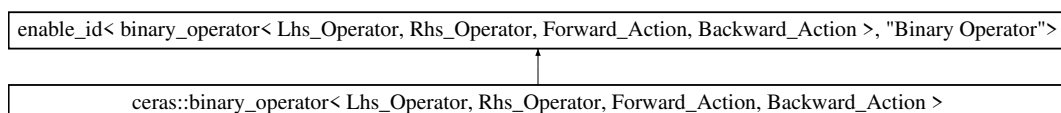
The documentation for this struct was generated from the following file:

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

6.4 ceras::binary_operator< Lhs_Operator, 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 Lhs_Operator [wrapped_lhs_operator_type](#)
- typedef Rhs_Operator [wrapped_rhs_operator_type](#)
- 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, std::function< void()> const &reset_action) noexcept
- auto [forward](#) ()
- void [backward](#) ([tensor_type](#) const &grad)
- void [reset_states](#) ()

Public Attributes

- Lhs_Operator [lhs_op_](#)
- Rhs_Operator [rhs_op_](#)
- Forward_Action [forward_action_](#)
- Backward_Action [backward_action_](#)
- std::function< void()> [reset_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
```

6.4.1.2 [wrapped_lhs_operator_type](#)

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
typedef Lhs_Operator ceras::binary\_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_
_Action >::wrapped\_lhs\_operator\_type
```


6.4.1.3 wrapped_rhs_operator_type

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
typedef Rhs_Operator ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_
_Action >::wrapped_rhs_operator_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,
    std::function< void()> const & reset_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]
```

6.4.3.3 reset_states()

```
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 >↵
::reset_states ( ) [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 reset_action_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
std::function<void()> ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action,
Backward_Action >::reset_action_
```

6.4.4.7 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_
```

6.4.4.8 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 T>`
auto `evaluate` (T const &inputs, T const &outputs, unsigned long batch_size=32)
- `template<Tensor T>`
auto `fit` (T const &inputs, T const &outputs, unsigned long batch_size, unsigned long epoch=1, int verbose=0, double validation_split=0.0)
- `template<Tensor T>`
auto `train_on_batch` (T const &input, T const &output)
- `template<Tensor T>`
auto `predict` (T const &input_tensor)
- `template<Expression Exp>`
auto `operator()` (Exp const &ex) const noexcept

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 );
}
```

6.5.4 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>
```

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_
```

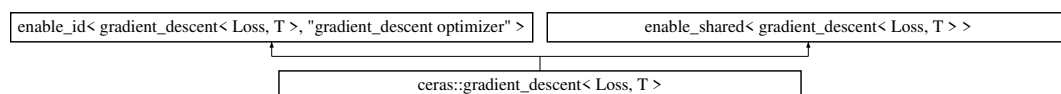
The documentation for this struct was generated from the following file:

- /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_
```

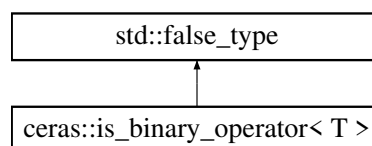
The documentation for this struct was generated from the following file:

- /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 >:



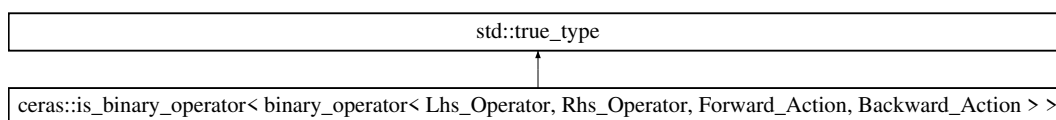
The documentation for this struct was generated from the following file:

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

6.9 ceras::is_binary_operator< binary_operator< Lhs_Operator, Rh_s_Operator, Forward_Action, Backward_Action > > Struct Template Reference

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

Inheritance diagram for ceras::is_binary_operator< binary_operator< Lhs_Operator, Rh_s_Operator, Forward_A_{ct}ion, Backward_A_{ct}ion > >:



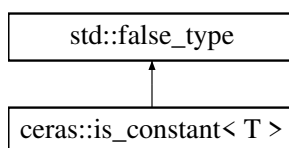
The documentation for this struct was generated from the following file:

- /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 >:



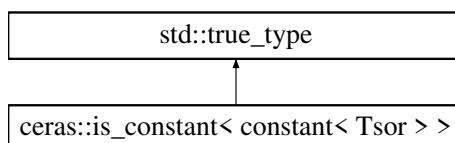
The documentation for this struct was generated from the following file:

- /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 > >:



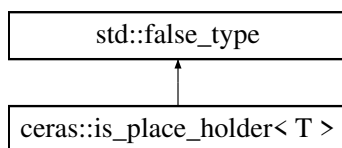
The documentation for this struct was generated from the following file:

- /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 >:



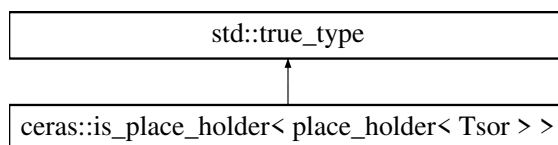
The documentation for this struct was generated from the following file:

- /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 > >:



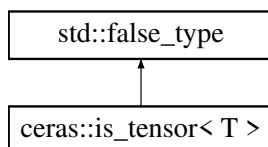
The documentation for this struct was generated from the following file:

- /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 >:



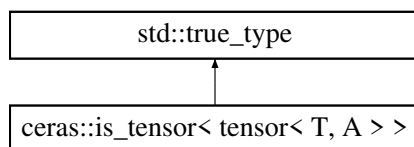
The documentation for this struct was generated from the following file:

- /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 > >`:



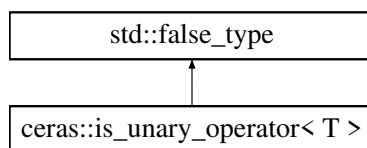
The documentation for this struct was generated from the following file:

- /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 >`:



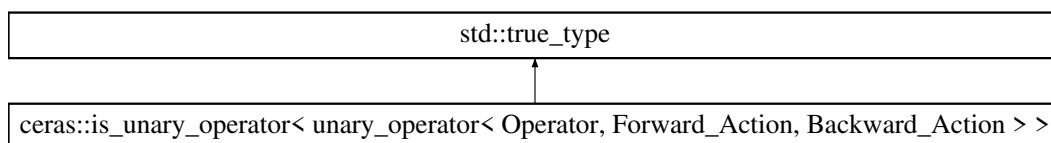
The documentation for this struct was generated from the following file:

- /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 > >`:



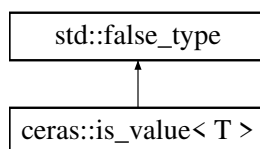
The documentation for this struct was generated from the following file:

- /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 >:



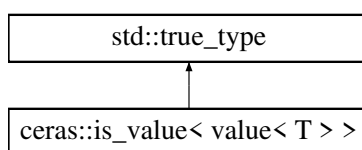
The documentation for this struct was generated from the following file:

- /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 > >:



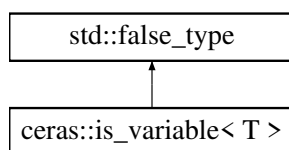
The documentation for this struct was generated from the following file:

- /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 >:



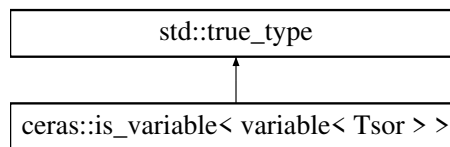
The documentation for this struct was generated from the following file:

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

6.21 `ceras::is_variable< variable< Tsor > >` Struct Template Reference

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

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



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

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

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

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

Public Types

- typedef Ph [input_layer_type](#)
- typedef Ex [output_layer_type](#)

Public Member Functions

- [input_layer_type](#) [input](#) () const noexcept
- [output_layer_type](#) [output](#) () const noexcept
- [model](#) ([input_layer_type](#) const &[place_holder](#), [output_layer_type](#) const &expression)
- template<Tensor Tsor>
auto [predict](#) (Tsor const &input_tensor)
- template<Expression Exp>
auto [operator\(\)](#) (Exp const &ex) const noexcept
- template<typename Loss , typename Optimizer >
auto [compile](#) (Loss const &l, Optimizer const &o)

Public Attributes

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

6.22.1 Detailed Description

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

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

Template Parameters

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

6.22.2 Member Typedef Documentation

6.22.2.1 input_layer_type

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

6.22.2.2 output_layer_type

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

6.22.3 Constructor & Destructor Documentation

6.22.3.1 model()

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

Parameters

<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 };
```

6.22.4 Member Function Documentation

6.22.4.1 compile()

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

Compile the model for training

Parameters

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

Returns

An instance of [compiled_model](#).

Example usage:

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

6.22.4.2 input()

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

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

6.22.4.3 operator>()

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

Generating a new expression by using the current model.

Parameters

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

Returns

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

Example code

```

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

```

6.22.4.4 output()

```

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

```

Returns the output layer of the model.

6.22.4.5 predict()

```

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

```

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

Parameters

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

Returns

The result this model predicts.

Example to predict

```

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

```

6.22.5 Member Data Documentation

6.22.5.1 expression_

```

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

```

output layer of the model.

6.22.5.2 place_holder_

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

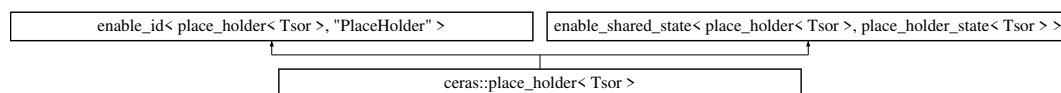
The documentation for this struct was generated from the following file:

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

6.23 ceras::place_holder< Tsor > Struct Template Reference

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

Inheritance diagram for ceras::place_holder< Tsor >:



Public Types

- typedef Tsor [tensor_type](#)

Public Member Functions

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

6.23.1 Member Typedef Documentation

6.23.1.1 tensor_type

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

6.23.2 Constructor & Destructor Documentation

6.23.2.1 place_holder() [1/4]

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

6.23.2.2 place_holder() [2/4]

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

6.23.2.3 place_holder() [3/4]

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

6.23.2.4 place_holder() [4/4]

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

6.23.3 Member Function Documentation

6.23.3.1 backward()

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

6.23.3.2 bind()

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

6.23.3.3 forward()

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

6.23.3.4 operator=() [1/2]

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

6.23.3.5 operator=() [2/2]

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

6.23.3.6 reset()

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

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

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

6.24 ceras::place_holder_state< Tsor > Struct Template Reference

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

Public Attributes

- Tsor [data_](#)
- std::vector< unsigned long > [shape_hint_](#)

6.24.1 Member Data Documentation

6.24.1.1 data_

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

6.24.1.2 shape_hint_

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

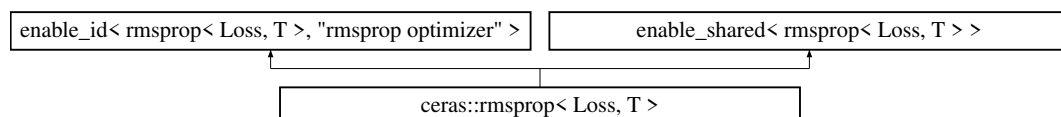
The documentation for this struct was generated from the following file:

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

6.25 ceras::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_`

6.25.1 Member Typedef Documentation

6.25.1.1 `tensor_type`

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

6.25.2 Constructor & Destructor Documentation

6.25.2.1 `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]
```

6.25.3 Member Function Documentation

6.25.3.1 `forward()`

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

6.25.4 Member Data Documentation

6.25.4.1 decay_

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

6.25.4.2 iterations_

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

6.25.4.3 learning_rate_

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

6.25.4.4 loss_

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

6.25.4.5 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

6.26 ceras::session< T_{src} > Struct Template Reference

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

Public Types

- typedef [place_holder< T_{src} > place_holder_type](#)
- typedef [variable< T_{src} > variable_type](#)
- typedef [variable_state< T_{src} > variable_state_type](#)

Public Member Functions

- [session](#) ()
- [session](#) ([session](#) const &)=delete
- [session](#) ([session](#) &&)=delete
- [session](#) & [operator=](#) ([session](#) const &)=delete
- [session](#) & [operator=](#) ([session](#) &&)=delete
- void [rebind](#) ([place_holder_type](#) &p_holder, Tsor const &[value](#))
- void [bind](#) ([place_holder_type](#) &p_holder, Tsor const &[value](#))
- void [remember](#) ([variable_type](#) const &v)
- template<typename Operation >
auto [run](#) (Operation &op) const
- template<typename Operation >
void [tap](#) (Operation &op) const
- void [deserialize](#) (std::string const &file_path)
- void [serialize](#) (std::string const &file_path) const
- void [save](#) (std::string const &file_path) const
- void [restore](#) (std::string const &file_path)
- [~session](#) ()

Public Attributes

- std::vector< [place_holder_type](#) > [place_holders_](#)
- std::unordered_map< int, [variable_type](#) > [variables_](#)

6.26.1 Member Typedef Documentation

6.26.1.1 [place_holder_type](#)

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

6.26.1.2 [variable_state_type](#)

```
template<Tensor Tsor>
typedef variable\_state<Tsor> ceras::session< Tsor >::variable\_state\_type
```

6.26.1.3 [variable_type](#)

```
template<Tensor Tsor>
typedef variable<Tsor> ceras::session< Tsor >::variable\_type
```

6.26.2 Constructor & Destructor Documentation

6.26.2.1 session() [1/3]

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

6.26.2.2 session() [2/3]

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

6.26.2.3 session() [3/3]

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

6.26.2.4 ~session()

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

6.26.3 Member Function Documentation

6.26.3.1 bind()

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

6.26.3.2 deserialize()

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

6.26.3.3 operator=() [1/2]

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

6.26.3.4 operator=() [2/2]

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

6.26.3.5 rebind()

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

6.26.3.6 remember()

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

6.26.3.7 restore()

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

6.26.3.8 run()

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

6.26.3.9 save()

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

6.26.3.10 serialize()

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

6.26.3.11 tap()

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

6.26.4 Member Data Documentation

6.26.4.1 place_holders_

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

6.26.4.2 variables_

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

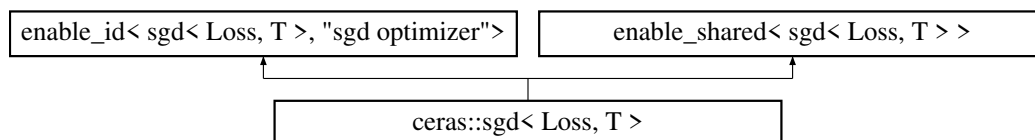
The documentation for this struct was generated from the following file:

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

6.27 ceras::sgd< Loss, T > Struct Template Reference

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

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



Public Types

- typedef [tensor](#)< T > [tensor_type](#)

Public Member Functions

- [sgd](#) (Loss &loss, std::size_t batch_size, T learning_rate=1.0e-1, T momentum=0.0, T decay=0.0, bool nesterov=false) noexcept
- void [forward](#) ()

Public Attributes

- Loss & [loss_](#)
- T [learning_rate_](#)
- T [momentum_](#)
- T [decay_](#)
- bool [nesterov_](#)
- unsigned long [iterations_](#)

6.27.1 Member Typedef Documentation

6.27.1.1 tensor_type

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

6.27.2 Constructor & Destructor Documentation

6.27.2.1 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]
```

6.27.3 Member Function Documentation

6.27.3.1 forward()

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

6.27.4 Member Data Documentation

6.27.4.1 decay_

```
template<typename Loss , typename T >
T ceras::sgd< Loss, T >::decay\_
```

6.27.4.2 iterations_

```
template<typename Loss , typename T >
unsigned long ceras::sgd< Loss, T >::iterations\_
```

6.27.4.3 learning_rate_

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

6.27.4.4 loss_

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

6.27.4.5 momentum_

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

6.27.4.6 nesterov_

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

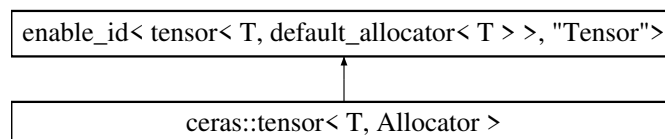
The documentation for this struct was generated from the following file:

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

6.28 ceras::tensor< T, Allocator > Struct Template Reference

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

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



Public Types

- typedef T [value_type](#)
- typedef Allocator [allocator](#)
- typedef std::vector< T, Allocator > [vector_type](#)
- typedef std::shared_ptr< [vector_type](#) > [shared_vector](#)
- typedef [tensor](#) [self_type](#)

Public Member Functions

- `tensor slice` (unsigned long m, unsigned long n) const noexcept
- `constexpr auto begin` () noexcept
- `constexpr auto begin` () const noexcept
- `constexpr auto cbegin` () const noexcept
- `constexpr auto end` () noexcept
- `constexpr auto end` () const noexcept
- `constexpr auto cend` () const noexcept
- `constexpr self_type & reset` (T val=T{0})
- `constexpr unsigned long ndim` () const noexcept
- `constexpr self_type & deep_copy` (self_type const &other)
- `constexpr self_type const deep_copy` () const
- `constexpr self_type const copy` () const
- `constexpr value_type & operator[]` (unsigned long idx)
- `constexpr value_type const & operator[]` (unsigned long idx) const
- `tensor` ()
- `constexpr tensor` (std::vector< unsigned long > const &shape, std::initializer_list< T > init, const Allocator &alloc=Allocator())
- `constexpr tensor` (std::vector< unsigned long > const &shape)
- `constexpr tensor` (std::vector< unsigned long > const &shape, T init)
- `constexpr tensor` (tensor const &other, unsigned long memory_offset)
- `constexpr tensor` (self_type const &other) noexcept
- `constexpr tensor` (self_type &&other) noexcept
- `constexpr self_type & operator=` (self_type const &other) noexcept
- `constexpr self_type & operator=` (self_type &&other) noexcept
- `constexpr std::vector< unsigned long > const & shape` () const noexcept
- `constexpr unsigned long size` () const noexcept
- `constexpr self_type & resize` (std::vector< unsigned long > const &new_shape)
- `constexpr self_type & reshape` (std::vector< unsigned long > const &new_shape)
- `constexpr self_type & shrink_to` (std::vector< unsigned long > const &new_shape)
- `constexpr self_type & creep_to` (unsigned long new_memory_offset)
- `constexpr bool empty` () const noexcept
- `constexpr value_type * data` () noexcept
- `constexpr const value_type * data` () const noexcept
- `template<typename Function >`
`constexpr self_type & map` (Function const &f)
- `constexpr self_type & operator+=` (self_type const &other)
- `constexpr self_type & operator+=` (value_type x)
- `constexpr self_type & operator-=` (self_type const &other)
- `constexpr self_type & operator-=` (value_type x)
- `constexpr self_type & operator*=` (self_type const &other)
- `constexpr self_type & operator*=` (value_type x)
- `constexpr self_type & operator/=` (self_type const &other)
- `constexpr self_type & operator/=` (value_type x)
- `constexpr self_type const operator-` () const
- `constexpr value_type as_scalar` () const noexcept
- `template<typename U >`
`constexpr auto as_type` () const noexcept

Public Attributes

- `std::vector< unsigned long > shape_`
- `unsigned long memory_offset_`
- `shared_vector vector_`

6.28.1 Member Typedef Documentation

6.28.1.1 allocator

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

6.28.1.2 self_type

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

6.28.1.3 shared_vector

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

6.28.1.4 value_type

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

6.28.1.5 vector_type

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

6.28.2 Constructor & Destructor Documentation

6.28.2.1 tensor() [1/7]

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

6.28.2.2 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]
```

6.28.2.3 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]
```

6.28.2.4 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]
```

6.28.2.5 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]
```

6.28.2.6 tensor() [6/7]

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

6.28.2.7 tensor() [7/7]

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

6.28.3 Member Function Documentation

6.28.3.1 as_scalar()

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

6.28.3.2 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]
```

6.28.3.3 begin() [1/2]

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

6.28.3.4 begin() [2/2]

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

6.28.3.5 cbegin()

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

6.28.3.6 cend()

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

6.28.3.7 copy()

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

6.28.3.8 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]
```

6.28.3.9 data() [1/2]

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

6.28.3.10 data() [2/2]

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

6.28.3.11 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]
```

6.28.3.12 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]
```

6.28.3.13 empty()

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

6.28.3.14 end() [1/2]

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

6.28.3.15 end() [2/2]

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

6.28.3.16 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]
```

6.28.3.17 ndim()

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

6.28.3.18 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]
```

6.28.3.19 `operator*=()` [2/2]

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

6.28.3.20 `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]
```

6.28.3.21 `operator+=()` [2/2]

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

6.28.3.22 `operator-()`

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

6.28.3.23 `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]
```

6.28.3.24 `operator-=()` [2/2]

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

6.28.3.25 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]
```

6.28.3.26 operator/=() [2/2]

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

6.28.3.27 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]
```

6.28.3.28 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]
```

6.28.3.29 operator[]() [1/2]

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

6.28.3.30 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]
```


6.28.3.31 reset()

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::reset (
    T val = T{0} ) [inline], [constexpr]
```

6.28.3.32 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]
```

6.28.3.33 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]
```

6.28.3.34 shape()

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

6.28.3.35 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]
```

6.28.3.36 size()

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

6.28.3.37 slice()

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

6.28.4 Member Data Documentation

6.28.4.1 memory_offset_

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

6.28.4.2 shape_

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

6.28.4.3 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](#)

6.29 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())>

6.29.1 Member Typedef Documentation

6.29.1.1 op_type

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

6.29.1.2 tensor_type

```
template<typename L , typename R >
using ceras::tensor_deduction< L, R >::tensor_type = std::remove_cv_t<decltype(std::declval<op_type>()).forward
```

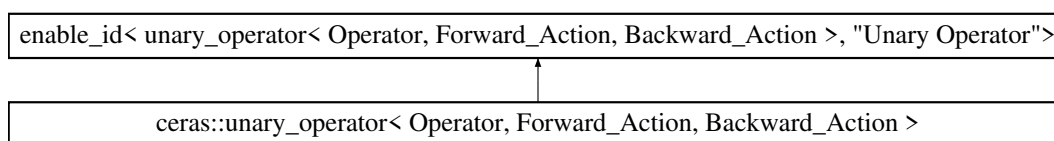
The documentation for this struct was generated from the following file:

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

6.30 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 Types

- typedef [Operator wrapped_operator_type](#)

Public Member Functions

- [unary_operator](#) ([Operator](#) const &op, [Forward_Action](#) const &forward_action, [Backward_Action](#) const &backward_action, std::function< void()> const &reset_action) noexcept
- auto [forward](#) ()
- void [backward](#) ([tensor_type](#) const &grad)
- void [reset_states](#) ()

Public Attributes

- [Operator](#) `op_`
- `Forward_Action` `forward_action_`
- `Backward_Action` `backward_action_`
- `std::function< void()>` `reset_action_`
- `decltype(std::declval< Forward_Action >()(std::declval< decltype(op_)>().forward()))` typedef `tensor_type`
- `tensor_type` `input_data_`
- `tensor_type` `output_data_`

6.30.1 Member Typedef Documentation

6.30.1.1 wrapped_operator_type

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
typedef Operator ceras::unary\_operator< Operator, Forward_Action, Backward_Action >::wrapped\_operator\_type
```

6.30.2 Constructor & Destructor Documentation

6.30.2.1 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,
    std::function< void()> const & reset_action ) [inline], [noexcept]
```

6.30.3 Member Function Documentation

6.30.3.1 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]
```

6.30.3.2 forward()

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

6.30.3.3 reset_states()

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

6.30.4 Member Data Documentation

6.30.4.1 backward_action_

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

6.30.4.2 forward_action_

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

6.30.4.3 input_data_

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

6.30.4.4 op_

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

6.30.4.5 output_data_

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

6.30.4.6 reset_action_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
std::function<void()> ceras::unary_operator< Operator, Forward_Action, Backward_Action >::reset_action_
```

6.30.4.7 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](#)

6.31 ceras::value< T > Struct Template Reference

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

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_](#)

6.31.1 Member Typedef Documentation

6.31.1.1 value_type

```
template<typename T >  
typedef T ceras::value< T >::value\_type
```

6.31.2 Constructor & Destructor Documentation

6.31.2.1 value() [1/4]

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

6.31.2.2 value() [2/4]

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

6.31.2.3 value() [3/4]

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

6.31.2.4 value() [4/4]

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

6.31.3 Member Function Documentation

6.31.3.1 backward()

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

6.31.3.2 forward()

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

6.31.3.3 operator=() [1/2]

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

6.31.3.4 operator=() [2/2]

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

6.31.4 Member Data Documentation

6.31.4.1 data_

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

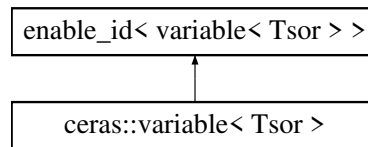
The documentation for this struct was generated from the following file:

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

6.32 ceras::variable< Tzor > Struct Template Reference

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

Inheritance diagram for ceras::variable< Tzor >:



Public Types

- typedef Tzor [tensor_type](#)

Public Member Functions

- [variable](#) (Tzor const &[data](#), bool trainable=true, bool stateful=false)
- [variable](#) ()=delete
- [variable](#) ([variable](#) const &other)=default
- [variable](#) ([variable](#) &&)=default
- [variable](#) & [operator=](#) ([variable](#) &&)=default
- [variable](#) & [operator=](#) ([variable](#) const &other)=default
- Tzor const [forward](#) () const
- void [backward](#) (auto const &grad)
- std::vector< std::size_t > [shape](#) () const noexcept
- std::vector< Tzor > & [contexts](#) ()
- std::vector< Tzor > [contexts](#) () const
- Tzor & [data](#) ()
- Tzor [data](#) () const
- Tzor & [gradient](#) ()
- Tzor [gradient](#) () const
- void [reset](#) ()
- void [reset_states](#) ()

Public Attributes

- std::shared_ptr< [variable_state](#)< Tzor > > [state_](#)
- bool [trainable_](#)
- bool [stateful_](#)

6.32.1 Member Typedef Documentation

6.32.1.1 tensor_type

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

6.32.2 Constructor & Destructor Documentation

6.32.2.1 variable() [1/4]

```
template<Tensor Tsor>
ceras::variable< Tsor >::variable (
    Tsor const & data,
    bool trainable = true,
    bool stateful = false ) [inline]
```

6.32.2.2 variable() [2/4]

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

6.32.2.3 variable() [3/4]

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

6.32.2.4 variable() [4/4]

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

6.32.3 Member Function Documentation

6.32.3.1 backward()

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

6.32.3.2 contexts() [1/2]

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

6.32.3.3 contexts() [2/2]

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

6.32.3.4 data() [1/2]

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

6.32.3.5 data() [2/2]

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

6.32.3.6 forward()

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

6.32.3.7 gradient() [1/2]

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

6.32.3.8 gradient() [2/2]

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

6.32.3.9 operator=() [1/2]

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

6.32.3.10 operator=() [2/2]

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

6.32.3.11 reset()

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

6.32.3.12 reset_states()

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

6.32.3.13 shape()

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

6.32.4 Member Data Documentation

6.32.4.1 state_

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

6.32.4.2 stateful_

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

6.32.4.3 trainable_

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

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

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

6.33 ceras::variable_state< Tzor > Struct Template Reference

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

Public Attributes

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

6.33.1 Member Data Documentation

6.33.1.1 contexts_

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

6.33.1.2 data_

```
template<Tensor Tsor>
Tsor ceras::variable\_state< Tsor >::data_
```

6.33.1.3 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](#)

6.34 [ceras::view_2d](#)< T > Struct Template Reference

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

Public Member Functions

- `template<typename A >`
`constexpr view_2d (tensor< T, A > &tsor, unsigned long row, unsigned long col, bool transposed=false) noexcept`
- `constexpr view_2d (T *data, unsigned long row, unsigned long col, bool transposed=false) noexcept`
- `constexpr view_2d (const T *data, unsigned long row, unsigned long col, bool transposed=false) noexcept`
- `constexpr T * operator\[\] (unsigned long index)`
- `constexpr const T * operator\[\] (unsigned long index) const`
- `constexpr auto shape () const noexcept`
- `constexpr unsigned long size () const noexcept`
- `constexpr T * data () noexcept`
- `constexpr const T * data () const noexcept`

Public Attributes

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

6.34.1 Constructor & Destructor Documentation

6.34.1.1 view_2d() [1/3]

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

6.34.1.2 view_2d() [2/3]

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

6.34.1.3 view_2d() [3/3]

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

6.34.2 Member Function Documentation**6.34.2.1 data()** [1/2]

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

6.34.2.2 data() [2/2]

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

6.34.2.3 operator[]() [1/2]

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

6.34.2.4 operator[]() [2/2]

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

6.34.2.5 shape()

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

6.34.2.6 size()

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

6.34.3 Member Data Documentation

6.34.3.1 col_

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

6.34.3.2 data_

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


6.34.3.3 row_

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

6.34.3.4 transposed_

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

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

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

6.35 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_](#)

6.35.1 Constructor & Destructor Documentation

6.35.1.1 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]
```

6.35.2 Member Function Documentation

6.35.2.1 operator[]() [1/2]

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

6.35.2.2 operator[]() [2/2]

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

6.35.3 Member Data Documentation

6.35.3.1 channel_

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

6.35.3.2 col_

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

6.35.3.3 data_

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

6.35.3.4 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

6.36 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.

6.36.1 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].

6.36.2 Constructor & Destructor Documentation

6.36.2.1 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.

6.36.3 Member Function Documentation

6.36.3.1 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];
```

6.36.3.2 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;
```

6.36.4 Member Data Documentation

6.36.4.1 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.

6.36.4.2 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.

6.36.4.3 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.

6.36.4.4 data_

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

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

6.36.4.5 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

Namespaces

- [ceras](#)

Variables

- constexpr unsigned long [ceras::version](#) = 20210418UL
- constexpr unsigned long [ceras::__version__](#) = version
- constexpr unsigned long [ceras::is_windows_platform](#) = 0
- constexpr unsigned long [ceras::debug_mode](#) = 1
- constexpr unsigned long [ceras::blas_mode](#) = 0
- constexpr unsigned long [ceras::cuda_mode](#) = 0
- int [ceras::visible_device](#) = 0
- unsigned long [ceras::cuda_gemm_threshold](#) = 0UL
- constexpr double [ceras::eps](#) = 1.0e-8
- int [ceras::learning_phase](#) = 1

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](#)

Functions

- auto [ceras::dataset::mnist::load_data](#) (std::string const &path=std::string{"./dataset/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 <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/keras.hpp File Reference

```
#include "../keras/layer.hpp"
#include "../keras/activation.hpp"
#include "../keras/application.hpp"
#include "../keras/callback.hpp"
#include "../keras/constraint.hpp"
#include "../keras/dataset.hpp"
#include "../keras/initializer.hpp"
#include "../keras/loss.hpp"
#include "../keras/metric.hpp"
#include "../keras/model.hpp"
#include "../keras/optimizer.hpp"
#include "../keras/regularizer.hpp"
#include "../keras/visualization.hpp"
```

7.8 /data/structured_folders/workspace/github.repo/ceras/include/layer.hpp File Reference

```
#include "../operation.hpp"
#include "../activation.hpp"
#include "../loss.hpp"
#include "../optimizer.hpp"
#include "../utils/better_assert.hpp"
```

Namespaces

- [ceras](#)

Functions

- auto [ceras::Input](#) ()
- auto [ceras::Conv2D](#) (unsigned long output_channels, std::vector< unsigned long > const &kernel_size, std::vector< unsigned long > const &input_shape, std::string const &padding="valid", std::vector< unsigned long > const &strides={1, 1})
- auto [ceras::Dense](#) (unsigned long output_size, unsigned long input_size)
- auto [ceras::BatchNormalization](#) (std::vector< unsigned long > const &shape, float threshold=0.95f)
- auto [ceras::Concatenate](#) (unsigned long axis=-1) noexcept
- auto [ceras::Add](#) () noexcept
- auto [ceras::Subtract](#) () noexcept
- auto [ceras::Multiply](#) () noexcept
- template<Expression Ex>
auto [ceras::ReLU](#) (Ex const &ex) noexcept
- auto [ceras::Softmax](#) () noexcept
- template<typename T = float>
auto [ceras::LeakyReLU](#) (T const factor=0.2) noexcept
- template<typename T = float>
auto [ceras::ELU](#) (T const factor=0.2) noexcept
- auto [ceras::Reshape](#) (std::vector< unsigned long > const &new_shape, bool include_batch_flag=true) noexcept

7.9 `/data/structured_↵` folders/workspace/github.repo/ceras/include/loss.hpp File Reference

```
#include "../operation.hpp"
#include "../tensor.hpp"
#include "../utils/debug.hpp"
```

Namespaces

- [ceras](#)

Functions

- template<Expression Lhs_Expression, Expression Rhx_Expression>
constexpr auto [ceras::mean_squared_logarithmic_error](#) (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhx_Expression>
constexpr auto [ceras::squared_loss](#) (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhx_Expression>
constexpr auto [ceras::mean_squared_error](#) (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept

- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto ceras::mse (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto ceras::abs_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto ceras::mean_absolute_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto ceras::mae (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto ceras::cross_entropy (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhs_Expression>`
`constexpr auto ceras::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`
- `auto ceras::MeanAbsoluteError`
- `auto ceras::Hinge`
- `auto ceras::CategoricalCrossentropy`

7.10 /data/structured_folders/workspace/github.repo/ceras/include/model.hpp File Reference

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

Classes

- `struct ceras::compiled_model< Model, Optimizer, Loss >`
- `struct ceras::model< Ex, Ph >`

Namespaces

- `ceras`

Functions

- `template<Expression Ex, Place_Holder Ph, Expression Ey>`
`auto ceras::replace_placeholder_with_expression (Ex const &ex, Ph const &old_place_holder, Ey const &new_expression)`
- `template<typename Model, typename Optimizer, typename Loss >`
`auto ceras::make_compiled_model (Model const &m, Loss const &l, Optimizer const &o)`

7.11 `/data/structured_↔` **folders/workspace/github.repo/ceras/include/operation.hpp** File **Reference**

```
#include "../includes.hpp"
#include "../place_holder.hpp"
#include "../variable.hpp"
#include "../constant.hpp"
#include "../value.hpp"
#include "../utils/range.hpp"
#include "../utils/debug.hpp"
#include "../config.hpp"
#include "../utils/context_cast.hpp"
#include "../utils/for_each.hpp"
#include "../utils/id.hpp"
#include "../utils/enable_shared.hpp"
```

Classes

- `struct ceras::unary_operator< Operator, Forward_Action, Backward_Action >`
- `struct ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >`
- `struct ceras::is_unary_operator< T >`
- `struct ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >`
- `struct ceras::is_binary_operator< T >`
- `struct ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > >`

Namespaces

- [ceras](#)

Functions

- `template<Expression Lhs_Expression, Expression Rhc_Expression>`
`constexpr auto ceras::plus (Lhs_Expression const &lhc_ex, Rhc_Expression const &rhc_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhc_Expression>`
`constexpr auto ceras::operator+ (Lhs_Expression const &lhc_ex, Rhc_Expression const &rhc_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rhc_Expression>`
`auto ceras::operator* (Lhs_Expression const &lhc_ex, Rhc_Expression const &rhc_ex) noexcept`
- `template<Expression Ex>`
`constexpr auto ceras::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::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
- 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<typename T = double>
requires std::floating_point< T > auto [ceras::normalization_instance](#) (T const momentum=0.98) noexcept
- template<typename T >
requires std::floating_point< T > auto [ceras::instance_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`

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.

7.12 /data/structured_↵

folders/workspace/github.repo/ceras/include/optimizer.hpp File Reference

```
#include "../config.hpp"
#include "../operation.hpp"
#include "../place_holder.hpp"
#include "../variable.hpp"
#include "../session.hpp"
#include "../utils/color.hpp"
#include "../utils/debug.hpp"
#include "../utils/id.hpp"
#include "../utils/enable_shared.hpp"
```


Classes

- struct `ceras::sgd< Loss, T >`
- struct `ceras::adagrad< Loss, T >`
- struct `ceras::rmsprop< Loss, T >`
- struct `ceras::adadelta< Loss, T >`
- struct `ceras::adam< Loss, T >`
- struct `ceras::gradient_descent< Loss, T >`

Namespaces

- `ceras`

Typedefs

- template<typename Loss , typename T >
using `ceras::ada_grad` = `adagrad< Loss, T >`
- template<typename Loss , typename T >
using `ceras::rms_prop` = `rmsprop< Loss, T >`
- template<typename Loss , typename T >
using `ceras::ada_delta` = `adadelta< Loss, T >`

Variables

- auto `ceras::Adam`
- auto `ceras::SGD`
- auto `ceras::Adagrad`
- auto `ceras::RMSprop`
- auto `ceras::Adadelta`

7.13 /data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp File Reference

```
#include "../includes.hpp"
#include "../tensor.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/debug.hpp"
#include "../utils/id.hpp"
#include "../utils/enable_shared.hpp"
#include "../utils/state.hpp"
```

Classes

- struct `ceras::place_holder_state< Tsor >`
- struct `ceras::place_holder< Tsor >`
- struct `ceras::is_place_holder< T >`
- struct `ceras::is_place_holder< place_holder< Tsor > >`

Namespaces

- [ceras](#)

Functions

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

Variables

- `template<class T >`
`constexpr bool ceras::is_place_holder_v = is_place_holder<T>::value`
- `template<typename T >`
`concept ceras::Place_Holder = is_place_holder_v<T>`

7.14 /data/structured_↵ folders/workspace/github.repo/ceras/include/recurrent_↵ operation.hpp File Reference

```
#include "../operation.hpp"
#include "../activation.hpp"
#include "../variable.hpp"
```

Namespaces

- [ceras](#)

Functions

- `template<Expression Lhs_Expression, Variable Rhs_Variable>`
`constexpr auto ceras::copy (Lhs_Expression const &lhs_ex, Rhs_Variable const &rhs_va) noexcept`

Variables

- `auto ceras::lstm`

7.15 /data/structured_↵/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::session< T, S >](#)

Namespaces

- [ceras](#)

Functions

- [template<Tensor T>](#)
[std::reference_wrapper< session< T, S > >](#) [ceras::get_default_session \(\)](#)

7.16 /data/structured_↵/workspace/github.repo/ceras/include/tensor.hpp File Reference

```
#include "../includes.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/range.hpp"
#include "../utils/stride_iterator.hpp"
#include "../utils/for_each.hpp"
#include "../utils/buffered_allocator.hpp"
#include "../utils/debug.hpp"
#include "../utils/id.hpp"
#include "../backend/cuda.hpp"
```

Classes

- struct [ceras::tensor< T, Allocator >](#)
- struct [ceras::is_tensor< T >](#)
- struct [ceras::is_tensor< tensor< T, A > >](#)
- struct [ceras::view_2d< T >](#)
- struct [ceras::view_3d< T >](#)
- struct [ceras::view_4d< T >](#)

Namespaces

- [ceras](#)

Typedefs

- `template<typename T >`
using [ceras::default_allocator](#) = `std::allocator< T >`
- `template<typename T >`
using [ceras::matrix](#) = `view_2d< T >`
- `template<typename T >`
using [ceras::cube](#) = `view_3d< T >`
- `template<typename T >`
using [ceras::tesseract](#) = `view_4d< T >`

Functions

- `template<typename T , typename A = default_allocator<T>>`
constexpr `tensor< T, A >` [ceras::as_tensor](#) (T val) noexcept
- `template<Tensor Tsr, typename CharT , typename Traits >`
`std::basic_ostream< CharT, Traits >` & [ceras::operator<<](#) (`std::basic_ostream< CharT, Traits >` &os_, Tsr const &tsr)
- `template<typename T >`
requires `std::floating_point< T >` void [ceras::gemm_cpu](#) (T const *A, bool a_transposed, T const *B, bool b_transposed, unsigned long m, unsigned long n, unsigned long k, T *C)
- void [ceras::update_cuda_gemm_threshold](#) ()
- `template<typename T >`
requires `std::floating_point< T >` void [ceras::gemm](#) (T const *A, bool a_transposed, T const *B, bool b_transposed, unsigned long m, unsigned long n, unsigned long k, T *C)
- `template<typename T >`
requires `std::floating_point< T >` void [ceras::gemm](#) (`view_2d< T >` const &x, `view_2d< T >` const &y, `view_2d< T >` &ans)
- `template<Tensor Tsr>`
Tsr [ceras::add](#) (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::operator+](#) (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::operator+](#) (typename Tsr::value_type const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::operator+](#) (Tsr const &lhs, typename Tsr::value_type const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::minus](#) (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::operator-](#) (Tsr const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::operator-](#) (typename Tsr::value_type const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::operator-](#) (Tsr const &lhs, typename Tsr::value_type const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::operator*](#) (typename Tsr::value_type const &lhs, Tsr const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::operator*](#) (Tsr const &lhs, typename Tsr::value_type const &rhs) noexcept
- `template<Tensor Tsr>`
Tsr [ceras::operator/](#) (Tsr const &lhs, typename Tsr::value_type const &rhs) noexcept

- template<Tensor Tsor>
Tsor [ceras::reshape](#) (Tsor const &ts, std::vector< unsigned long > const &new_shape)
- template<Tensor Tsor>
void [ceras::multiply](#) (Tsor const &lhs, Tsor const &rhs, Tsor &ans) noexcept
- template<Tensor Tsor>
Tsor [ceras::multiply](#) (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
Tsor [ceras::operator*](#) (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
Tsor [ceras::elementwise_product](#) (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
Tsor [ceras::hadamard_product](#) (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
Tsor [ceras::elementwise_divide](#) (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
Tsor [ceras::repeat](#) (Tsor const &tsor, unsigned long n)
- template<Tensor Tsor>
Tsor [ceras::reduce_sum](#) (Tsor const &tsor)
- template<Tensor Tsor>
Tsor [ceras::reduce_mean](#) (Tsor const &tsor)
- template<Tensor Tsor>
Tsor [ceras::clip](#) (Tsor &tsor, typename Tsor::value_type lower=0, typename Tsor::value_type upper=1)
- template<Tensor Tsor>
Tsor [ceras::squeeze](#) (Tsor const &tsor)
- template<typename T, typename A = default_allocator<T>>
tensor< T, A > [ceras::randn](#) (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1})
- template<typename T, typename A = default_allocator<T>>
tensor< T, A > [ceras::truncated_normal](#) (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1}, T lower=T{0}, T upper=T{1})
- template<typename T, typename A = default_allocator<T>>
tensor< T, A > [ceras::random](#) (std::vector< unsigned long > const &shape, T min=T{0}, T max=T{1})
- template<Tensor Tsor>
Tsor [ceras::random_like](#) (Tsor const &tsor, typename Tsor::value_type min=0, typename Tsor::value_type max=1)
- template<Tensor Tsor>
Tsor [ceras::randn_like](#) (Tsor const &tsor, typename Tsor::value_type mean=0, typename Tsor::value_type stddev=1)
- template<typename T, typename A = default_allocator<T>>
tensor< T, A > [ceras::glorot_uniform](#) (std::initializer_list< unsigned long > shape)
- template<Tensor Tsor>
Tsor [ceras::deep_copy](#) (Tsor const &tsor)
- template<Tensor Tsor>
Tsor [ceras::copy](#) (Tsor const &tsor)
- template<Tensor Tsor>
Tsor [ceras::concatenate](#) (Tsor const &lhs, Tsor const &rhs, unsigned long axis=0) noexcept
- template<Tensor Tsor>
Tsor [ceras::repmat](#) (Tsor const &tsor, unsigned long row_rep, unsigned long col_rep)
- template<Tensor Tsor>
constexpr bool [ceras::empty](#) (Tsor const &tsor) noexcept
- template<typename T, typename A = default_allocator<T>>
constexpr tensor< T, A > [ceras::zeros](#) (std::vector< unsigned long > const &shape)
- template<Tensor Tsor>
constexpr Tsor [ceras::zeros_like](#) (Tsor const &tsor)
- template<typename T, typename A = default_allocator<T>>
constexpr tensor< T, A > [ceras::ones](#) (std::vector< unsigned long > const &shape)
- template<Tensor Tsor>
constexpr Tsor [ceras::ones_like](#) (Tsor const &tsor)

- template<Tensor 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>
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>`

7.17 /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>`

7.18 /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::variable< Tsor >](#)
- struct [ceras::is_variable< T >](#)
- struct [ceras::is_variable< variable< Tsor > >](#)

Namespaces

- [ceras](#)

Functions

- template<Tensor Tsor>
std::reference_wrapper< session< Tsor > > [ceras::get_default_session](#) ()
- template<Variable Var>
bool [ceras::operator==](#) (Var const &lhs, Var const &rhs) noexcept

Variables

- template<class T >
constexpr bool [ceras::is_variable_v](#) = is_variable<T>::value
- template<typename T >
concept [ceras::Variable](#) = is_variable_v<T>

Index

/data/structured_folders/workspace/github.repo/ceras/include/activation.hpp, 45
125
adadelta
/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp, 45
126
Adagrad
/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp, 46
126
adagrad
/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp, 46
127
Adam
/data/structured_folders/workspace/github.repo/ceras/include/data.hpp, 46
127
adam
/data/structured_folders/workspace/github.repo/ceras/include/include.hpp, 46
128
Adadelta
/data/structured_folders/workspace/github.repo/ceras/include/keras.hpp, 48
129
add
/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp, 49
129
allocator
/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp, 49
130
amax
/data/structured_folders/workspace/github.repo/ceras/include/model.hpp, 49
131
amin
/data/structured_folders/workspace/github.repo/ceras/include/operator.hpp, 49
132
amsgrad
/data/structured_folders/workspace/github.repo/ceras/include/operator.hpp, 49
134
as_scalar
/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp, 49
135
as_tensor
/data/structured_folders/workspace/github.repo/ceras/include/recurrent_operation.hpp, 49
136
as_type
/data/structured_folders/workspace/github.repo/ceras/include/session.hpp, 49
137
average_pooling_2d
/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp, 49
137
backward
/data/structured_folders/workspace/github.repo/ceras/include/value.hpp, 49
141
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp, 49
141
__version__
 ceras, 45
~session
 ceras::session< Tzor >, 89
abs
 ceras, 18
abs_loss
 ceras, 18
ada_delta
 ceras, 17
ada_grad
 ceras, 17
Adadelta
 ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >, 63
 ceras::constant< Tzor >, 70
 ceras::place_holder< Tzor >, 83
 ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 106
 ceras::value< T >, 110
 ceras::variable< Tzor >, 112
backward_action
 ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >, 64
 ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 107
batch_normalization
 ceras, 20
batch_size
 ceras::view_4d< T >, 124

BatchNormalization
 ceras, 20
 begin
 ceras::tensor< T, Allocator >, 98
 beta_1_
 ceras::adam< Loss, T >, 60
 beta_2_
 ceras::adam< Loss, T >, 61
 Binary_Operator
 ceras, 46
 binary_operator
 ceras::binary_operator< Lhs_Operator, Rhs_Operator,
 Forward_Action, Backward_Action >, 63
 bind
 ceras::place_holder< Tsr >, 83
 ceras::session< Tsr >, 89
 blas_mode
 ceras, 46
 CategoricalCrossentropy
 ceras, 46
 cbegin
 ceras::tensor< T, Allocator >, 98
 cend
 ceras::tensor< T, Allocator >, 98
 ceras, 9
 __version__, 45
 abs, 18
 abs_loss, 18
 ada_delta, 17
 ada_grad, 17
 Adadelta, 45
 Adagrad, 46
 Adam, 46
 Add, 18
 add, 19
 amax, 19
 amin, 19
 as_tensor, 19
 average_pooling_2d, 19
 batch_normalization, 20
 BatchNormalization, 20
 Binary_Operator, 46
 blas_mode, 46
 CategoricalCrossentropy, 46
 clip, 20
 concat, 20
 Concatenate, 21
 concatenate, 21
 Constant, 47
 Conv2D, 21
 conv2d, 22
 copy, 22
 cross_entropy, 22
 cross_entropy_loss, 22
 cube, 17
 cuda_gemm_threshold, 47
 cuda_mode, 47
 debug_mode, 47
 deep_copy, 23
 default_allocator, 17
 Dense, 23
 drop_out, 23
 elementwise_divide, 23
 elementwise_product, 23
 ELU, 24
 elu, 24
 empty, 24
 eps, 47
 exp, 24
 exponential, 24
 Expression, 47
 flatten, 24
 gelu, 25
 gemm, 25
 gemm_cpu, 25
 get_default_session, 25
 glorot_uniform, 26
 hadamard_product, 26
 hard_sigmoid, 26
 has_inf, 26
 has_nan, 26
 Hinge, 48
 hinge_loss, 27
 identity, 27
 img2col, 27
 Input, 27
 instance_normalization, 27
 is_binary_operator_v, 48
 is_constant_v, 48
 is_place_holder_v, 48
 is_tensor_v, 48
 is_unary_operator_v, 48
 is_valid, 27
 is_value_v, 49
 is_variable_v, 49
 is_windows_platform, 49
 leaky_relu, 28
 LeakyReLU, 28
 learning_phase, 49
 linspace, 28
 load_tensor, 28
 log, 28
 lstm, 49
 mae, 28
 make_binary_operator, 49
 make_compiled_model, 29
 make_unary_operator, 50
 matrix, 17
 max, 29
 max_pooling_2d, 29
 maximum, 29
 mean, 29, 30
 mean_absolute_error, 30
 mean_reduce, 30
 mean_squared_error, 30
 mean_squared_logarithmic_error, 30

- MeanAbsoluteError, [50](#)
- MeanSquaredError, [50](#)
- min, [30](#), [31](#)
- minus, [31](#)
- mse, [31](#)
- Multiply, [31](#)
- multiply, [32](#)
- negative, [32](#)
- negative_relu, [32](#)
- norm, [32](#)
- normalization_batch, [33](#)
- normalization_instance, [33](#)
- ones, [33](#)
- ones_like, [33](#)
- Operator, [50](#)
- operator!=, [33](#)
- operator<, [36](#)
- operator<<, [36](#)
- operator<=, [36](#)
- operator>, [36](#)
- operator>=, [37](#)
- operator*, [33](#), [34](#)
- operator+, [34](#), [35](#)
- operator-, [35](#)
- operator/, [35](#)
- operator==, [36](#)
- Place_Holder, [51](#)
- plus, [37](#)
- randn, [37](#)
- randn_like, [37](#)
- random, [37](#)
- random_generator, [51](#)
- random_like, [38](#)
- random_normal_like, [38](#)
- random_seed, [51](#)
- read_tensor, [38](#)
- reduce, [38](#)
- reduce_mean, [39](#)
- reduce_sum, [39](#)
- ReLU, [39](#)
- relu, [39](#)
- repeat, [40](#)
- replace_placeholder_with_expression, [40](#)
- repmat, [40](#)
- Reshape, [40](#)
- reshape, [41](#)
- rms_prop, [18](#)
- RMSprop, [51](#)
- save_tensor, [41](#)
- selu, [41](#)
- SGD, [51](#)
- sigmoid, [41](#)
- Softmax, [41](#)
- softmax, [42](#)
- softplus, [42](#)
- softsign, [42](#)
- square, [42](#)
- squared_loss, [42](#)
- squeeze, [43](#)
- standard_deviation, [43](#)
- Subtract, [43](#)
- sum, [43](#)
- sum_reduce, [44](#)
- tanh, [44](#)
- Tensor, [51](#)
- tesseract, [18](#)
- transpose, [44](#)
- truncated_normal, [44](#)
- Unary_Operator, [52](#)
- up_sampling_2d, [44](#)
- update_cuda_gemm_threshold, [44](#)
- Value, [52](#)
- Variable, [52](#)
- variance, [45](#)
- version, [52](#)
- visible_device, [52](#)
- write_tensor, [45](#)
- zeros, [45](#)
- zeros_like, [45](#)
- ceras::adadelta< Loss, T >, [55](#)
 - adadelta, [56](#)
 - forward, [56](#)
 - iterations_, [56](#)
 - learning_rate_, [56](#)
 - loss_, [56](#)
 - rho_, [57](#)
 - tensor_type, [55](#)
- ceras::adagrad< Loss, T >, [57](#)
 - adagrad, [58](#)
 - decay_, [58](#)
 - forward, [58](#)
 - iterations_, [58](#)
 - learning_rate_, [58](#)
 - loss_, [59](#)
 - tensor_type, [57](#)
- ceras::adam< Loss, T >, [59](#)
 - adam, [60](#)
 - amsgrad_, [60](#)
 - beta_1_, [60](#)
 - beta_2_, [61](#)
 - forward, [60](#)
 - iterations_, [61](#)
 - learning_rate_, [61](#)
 - loss_, [61](#)
 - tensor_type, [60](#)
- ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >, [61](#)
 - backward, [63](#)
 - backward_action_, [64](#)
 - binary_operator, [63](#)
 - forward, [63](#)
 - forward_action_, [64](#)
 - lhs_input_data_, [64](#)
 - lhs_op_, [64](#)
 - output_data_, [64](#)
 - reset_action_, [65](#)

- reset_states, 63
- rhs_input_data_, 65
- rhs_op_, 65
- tensor_type, 62
- wrapped_lhs_operator_type, 62
- wrapped_rhs_operator_type, 62
- ceras::compiled_model< Model, Optimizer, Loss >, 65
 - compiled_model, 66
 - compiled_optimizer_, 69
 - evaluate, 67
 - fit, 67
 - ground_truth_place_holder_, 69
 - input_place_holder_, 69
 - io_layer_type, 66
 - loss_, 69
 - model_, 69
 - operator(), 68
 - optimizer_, 69
 - optimizer_type, 70
 - predict, 68
 - train_on_batch, 68
- ceras::constant< Tzor >, 70
 - backward, 70
 - constant, 70
 - data_, 71
 - forward, 71
 - shape, 71
- ceras::dataset, 52
- ceras::dataset::mnist, 53
 - load_data, 53
- ceras::gradient_descent< Loss, T >, 71
 - forward, 72
 - gradient_descent, 72
 - learning_rate_, 72
 - loss_, 73
 - momentum_, 73
 - tensor_type, 72
- ceras::is_binary_operator< binary_operator< Lhs_Operator,
 - Rhs_Operator, Forward_Action, Backward_Action > >, 74
- ceras::is_binary_operator< T >, 73
- ceras::is_constant< constant< Tzor > >, 74
- ceras::is_constant< T >, 74
- ceras::is_place_holder< place_holder< Tzor > >, 75
- ceras::is_place_holder< T >, 75
- ceras::is_tensor< T >, 75
- ceras::is_tensor< tensor< T, A > >, 76
- ceras::is_unary_operator< T >, 76
- ceras::is_unary_operator< unary_operator< Operator,
 - Forward_Action, Backward_Action > >, 76
- ceras::is_value< T >, 77
- ceras::is_value< value< T > >, 77
- ceras::is_variable< T >, 77
- ceras::is_variable< variable< Tzor > >, 78
- ceras::model< Ex, Ph >, 78
 - compile, 79
 - expression_, 81
 - input, 80
 - input_layer_type, 79
 - model, 79
 - operator(), 80
 - output, 81
 - output_layer_type, 79
 - place_holder_, 81
 - predict, 81
- ceras::place_holder< Tzor >, 82
 - backward, 83
 - bind, 83
 - forward, 84
 - operator=, 84
 - place_holder, 83
 - reset, 84
 - tensor_type, 82
- ceras::place_holder_state< Tzor >, 84
 - data_, 85
 - shape_hint_, 85
- ceras::rmsprop< Loss, T >, 85
 - decay_, 86
 - forward, 86
 - iterations_, 87
 - learning_rate_, 87
 - loss_, 87
 - rho_, 87
 - rmsprop, 86
 - tensor_type, 86
- ceras::session< Tzor >, 87
 - ~session, 89
 - bind, 89
 - deserialize, 89
 - operator=, 90
 - place_holder_type, 88
 - place_holders_, 91
 - rebind, 90
 - remember, 90
 - restore, 90
 - run, 90
 - save, 91
 - serialize, 91
 - session, 89
 - tap, 91
 - variable_state_type, 88
 - variable_type, 88
 - variables_, 91
- ceras::sgd< Loss, T >, 92
 - decay_, 93
 - forward, 93
 - iterations_, 93
 - learning_rate_, 93
 - loss_, 94
 - momentum_, 94
 - nesterov_, 94
 - sgd, 93
 - tensor_type, 92
- ceras::tensor< T, Allocator >, 94
 - allocator, 96
 - as_scalar, 98

- as_type, 98
- begin, 98
- cbegin, 98
- cend, 98
- copy, 98
- creep_to, 99
- data, 99
- deep_copy, 99
- empty, 99
- end, 100
- map, 100
- memory_offset_, 104
- ndim, 100
- operator*=: 100
- operator+=, 101
- operator-, 101
- operator-=, 101
- operator/=: 101, 102
- operator=, 102
- operator[], 102
- reset, 102
- reshape, 103
- resize, 103
- self_type, 96
- shape, 103
- shape_, 104
- shared_vector, 96
- shrink_to, 103
- size, 103
- slice, 103
- tensor, 96, 97
- value_type, 96
- vector_, 104
- vector_type, 96
- ceras::tensor_deduction< L, R >, 104
 - op_type, 105
 - tensor_type, 105
- ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 105
 - backward, 106
 - backward_action_, 107
 - forward, 106
 - forward_action_, 107
 - input_data_, 107
 - op_, 107
 - output_data_, 107
 - reset_action_, 108
 - reset_states, 107
 - tensor_type, 108
 - unary_operator, 106
 - wrapped_operator_type, 106
- ceras::value< T >, 108
 - backward, 110
 - data_, 110
 - forward, 110
 - operator=, 110
 - value, 109
 - value_type, 109
- ceras::variable< T, Sor >, 111
 - backward, 112
 - contexts, 113
 - data, 113
 - forward, 113
 - gradient, 113
 - operator=, 114
 - reset, 114
 - reset_states, 114
 - shape, 114
 - state_, 114
 - stateful_, 115
 - tensor_type, 111
 - trainable_, 115
 - variable, 112
- ceras::variable_state< T, Sor >, 115
 - contexts_, 115
 - data_, 115
 - gradient_, 116
- ceras::view_2d< T >, 116
 - col_, 118
 - data, 117
 - data_, 118
 - operator[], 117, 118
 - row_, 118
 - shape, 118
 - size, 118
 - transposed_, 119
 - view_2d, 116, 117
- ceras::view_3d< T >, 119
 - channel_, 120
 - col_, 120
 - data_, 120
 - operator[], 120
 - row_, 120
 - view_3d, 119
- ceras::view_4d< T >, 121
 - batch_size_, 124
 - channel_, 124
 - col_, 124
 - data_, 124
 - operator[], 123
 - row_, 124
 - view_4d, 121
- channel_
 - ceras::view_3d< T >, 120
 - ceras::view_4d< T >, 124
- clip
 - ceras, 20
- col_
 - ceras::view_2d< T >, 118
 - ceras::view_3d< T >, 120
 - ceras::view_4d< T >, 124
- compile
 - ceras::model< Ex, Ph >, 79
- compiled_model
 - ceras::compiled_model< Model, Optimizer, Loss >, 66

- compiled_optimizer_
 - ceras::compiled_model< Model, Optimizer, Loss >, 69
- concat
 - ceras, 20
- Concatenate
 - ceras, 21
- concatenate
 - ceras, 21
- Constant
 - ceras, 47
- constant
 - ceras::constant< T, Allocator >, 70
- contexts
 - ceras::variable< T, Allocator >, 113
- contexts_
 - ceras::variable_state< T, Allocator >, 115
- Conv2D
 - ceras, 21
- conv2d
 - ceras, 22
- copy
 - ceras, 22
 - ceras::tensor< T, Allocator >, 98
- creep_to
 - ceras::tensor< T, Allocator >, 99
- cross_entropy
 - ceras, 22
- cross_entropy_loss
 - ceras, 22
- cube
 - ceras, 17
- cuda_gemm_threshold
 - ceras, 47
- cuda_mode
 - ceras, 47
- data
 - ceras::tensor< T, Allocator >, 99
 - ceras::variable< T, Allocator >, 113
 - ceras::view_2d< T, Allocator >, 117
- data_
 - ceras::constant< T, Allocator >, 71
 - ceras::place_holder_state< T, Allocator >, 85
 - ceras::value< T, Allocator >, 110
 - ceras::variable_state< T, Allocator >, 115
 - ceras::view_2d< T, Allocator >, 118
 - ceras::view_3d< T, Allocator >, 120
 - ceras::view_4d< T, Allocator >, 124
- debug_mode
 - ceras, 47
- decay_
 - ceras::adagrad< Loss, T, Allocator >, 58
 - ceras::rmsprop< Loss, T, Allocator >, 86
 - ceras::sgd< Loss, T, Allocator >, 93
- deep_copy
 - ceras, 23
 - ceras::tensor< T, Allocator >, 99
- default_allocator
 - ceras, 17
- Dense
 - ceras, 23
- deserialize
 - ceras::session< T, Allocator >, 89
- drop_out
 - ceras, 23
- elementwise_divide
 - ceras, 23
- elementwise_product
 - ceras, 23
- ELU
 - ceras, 24
- elu
 - ceras, 24
- empty
 - ceras, 24
 - ceras::tensor< T, Allocator >, 99
- end
 - ceras::tensor< T, Allocator >, 100
- eps
 - ceras, 47
- evaluate
 - ceras::compiled_model< Model, Optimizer, Loss >, 67
- exp
 - ceras, 24
- exponential
 - ceras, 24
- Expression
 - ceras, 47
- expression_
 - ceras::model< Ex, Ph >, 81
- fit
 - ceras::compiled_model< Model, Optimizer, Loss >, 67
- flatten
 - ceras, 24
- forward
 - ceras::adadelta< Loss, T, Allocator >, 56
 - ceras::adagrad< Loss, T, Allocator >, 58
 - ceras::adam< Loss, T, Allocator >, 60
 - ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >, 63
 - ceras::constant< T, Allocator >, 71
 - ceras::gradient_descent< Loss, T, Allocator >, 72
 - ceras::place_holder< T, Allocator >, 84
 - ceras::rmsprop< Loss, T, Allocator >, 86
 - ceras::sgd< Loss, T, Allocator >, 93
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 106
 - ceras::value< T, Allocator >, 110
 - ceras::variable< T, Allocator >, 113
- forward_action_
 - ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >, 64

- ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 107
- gelu
 - ceras, 25
- gemv
 - ceras, 25
- gemv_cpu
 - ceras, 25
- get_default_session
 - ceras, 25
- glorot_uniform
 - ceras, 26
- gradient
 - ceras::variable< Tsor >, 113
- gradient_
 - ceras::variable_state< Tsor >, 116
- gradient_descent
 - ceras::gradient_descent< Loss, T >, 72
- ground_truth_place_holder_
 - ceras::compiled_model< Model, Optimizer, Loss >, 69
- hadamard_product
 - ceras, 26
- hard_sigmoid
 - ceras, 26
- has_inf
 - ceras, 26
- has_nan
 - ceras, 26
- Hinge
 - ceras, 48
- hinge_loss
 - ceras, 27
- identity
 - ceras, 27
- img2col
 - ceras, 27
- includes.hpp
 - STB_IMAGE_IMPLEMENTATION, 128
 - STB_IMAGE_RESIZE_IMPLEMENTATION, 129
 - STB_IMAGE_WRITE_IMPLEMENTATION, 129
- Input
 - ceras, 27
- input
 - ceras::model< Ex, Ph >, 80
- input_data_
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 107
- input_layer_type
 - ceras::model< Ex, Ph >, 79
- input_place_holder_
 - ceras::compiled_model< Model, Optimizer, Loss >, 69
- instance_normalization
 - ceras, 27
- io_layer_type
 - ceras::compiled_model< Model, Optimizer, Loss >, 66
- is_binary_operator_v
 - ceras, 48
- is_constant_v
 - ceras, 48
- is_place_holder_v
 - ceras, 48
- is_tensor_v
 - ceras, 48
- is_unary_operator_v
 - ceras, 48
- is_valid
 - ceras, 27
- is_value_v
 - ceras, 49
- is_variable_v
 - ceras, 49
- is_windows_platform
 - ceras, 49
- iterations_
 - ceras::adadelta< Loss, T >, 56
 - ceras::adagrad< Loss, T >, 58
 - ceras::adam< Loss, T >, 61
 - ceras::rmsprop< Loss, T >, 87
 - ceras::sgd< Loss, T >, 93
- leaky_relu
 - ceras, 28
- LeakyReLU
 - ceras, 28
- learning_phase
 - ceras, 49
- learning_rate_
 - ceras::adadelta< Loss, T >, 56
 - ceras::adagrad< Loss, T >, 58
 - ceras::adam< Loss, T >, 61
 - ceras::gradient_descent< Loss, T >, 72
 - ceras::rmsprop< Loss, T >, 87
 - ceras::sgd< Loss, T >, 93
- lhs_input_data_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 64
- lhs_op_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 64
- linspace
 - ceras, 28
- load_data
 - ceras::dataset::mnist, 53
- load_tensor
 - ceras, 28
- log
 - ceras, 28
- loss_
 - ceras::adadelta< Loss, T >, 56
 - ceras::adagrad< Loss, T >, 59
 - ceras::adam< Loss, T >, 61

- ceras::compiled_model< Model, Optimizer, Loss >, 69
 - ceras::gradient_descent< Loss, T >, 73
 - ceras::rmsprop< Loss, T >, 87
 - ceras::sgd< Loss, T >, 94
- lstm
 - ceras, 49
- mae
 - ceras, 28
- make_binary_operator
 - ceras, 49
- make_compiled_model
 - ceras, 29
- make_unary_operator
 - ceras, 50
- map
 - ceras::tensor< T, Allocator >, 100
- matrix
 - ceras, 17
- max
 - ceras, 29
- max_pooling_2d
 - ceras, 29
- maximum
 - ceras, 29
- mean
 - ceras, 29, 30
- mean_absolute_error
 - ceras, 30
- mean_reduce
 - ceras, 30
- mean_squared_error
 - ceras, 30
- mean_squared_logarithmic_error
 - ceras, 30
- MeanAbsoluteError
 - ceras, 50
- MeanSquaredError
 - ceras, 50
- memory_offset_
 - ceras::tensor< T, Allocator >, 104
- min
 - ceras, 30, 31
- minus
 - ceras, 31
- model
 - ceras::model< Ex, Ph >, 79
- model_
 - ceras::compiled_model< Model, Optimizer, Loss >, 69
- momentum_
 - ceras::gradient_descent< Loss, T >, 73
 - ceras::sgd< Loss, T >, 94
- mse
 - ceras, 31
- Multiply
 - ceras, 31
- multiply
 - ceras, 32
- ndim
 - ceras::tensor< T, Allocator >, 100
- negative
 - ceras, 32
- negative_relu
 - ceras, 32
- nesterov_
 - ceras::sgd< Loss, T >, 94
- norm
 - ceras, 32
- normalization_batch
 - ceras, 33
- normalization_instance
 - ceras, 33
- ones
 - ceras, 33
- ones_like
 - ceras, 33
- op_
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 107
- op_type
 - ceras::tensor_deduction< L, R >, 105
- Operator
 - ceras, 50
- operator!=
 - ceras, 33
- operator<
 - ceras, 36
- operator<<
 - ceras, 36
- operator<=
 - ceras, 36
- operator>
 - ceras, 36
- operator>=
 - ceras, 37
- operator*
 - ceras, 33, 34
- operator*=
 - ceras::tensor< T, Allocator >, 100
- operator()
 - ceras::compiled_model< Model, Optimizer, Loss >, 68
 - ceras::model< Ex, Ph >, 80
- operator+
 - ceras, 34, 35
- operator+=
 - ceras::tensor< T, Allocator >, 101
- operator-
 - ceras, 35
 - ceras::tensor< T, Allocator >, 101
- operator-=
 - ceras::tensor< T, Allocator >, 101
- operator/
 - ceras, 35

- operator/=
 - ceras::tensor< T, Allocator >, [101](#), [102](#)
- operator=
 - ceras::place_holder< T, Allocator >, [84](#)
 - ceras::session< T, Allocator >, [90](#)
 - ceras::tensor< T, Allocator >, [102](#)
 - ceras::value< T >, [110](#)
 - ceras::variable< T, Allocator >, [114](#)
- operator==
 - ceras, [36](#)
- operator[]
 - ceras::tensor< T, Allocator >, [102](#)
 - ceras::view_2d< T >, [117](#), [118](#)
 - ceras::view_3d< T >, [120](#)
 - ceras::view_4d< T >, [123](#)
- optimizer_
 - ceras::compiled_model< Model, Optimizer, Loss >, [69](#)
- optimizer_type
 - ceras::compiled_model< Model, Optimizer, Loss >, [70](#)
- output
 - ceras::model< Ex, Ph >, [81](#)
- output_data_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [64](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [107](#)
- output_layer_type
 - ceras::model< Ex, Ph >, [79](#)
- Place_Holder
 - ceras, [51](#)
- place_holder
 - ceras::place_holder< T, Allocator >, [83](#)
- place_holder_
 - ceras::model< Ex, Ph >, [81](#)
- place_holder_type
 - ceras::session< T, Allocator >, [88](#)
- place_holders_
 - ceras::session< T, Allocator >, [91](#)
- plus
 - ceras, [37](#)
- predict
 - ceras::compiled_model< Model, Optimizer, Loss >, [68](#)
 - ceras::model< Ex, Ph >, [81](#)
- randn
 - ceras, [37](#)
- randn_like
 - ceras, [37](#)
- random
 - ceras, [37](#)
- random_generator
 - ceras, [51](#)
- random_like
 - ceras, [38](#)
- random_normal_like
 - ceras, [38](#)
- random_seed
 - ceras, [51](#)
- read_tensor
 - ceras, [38](#)
- rebind
 - ceras::session< T, Allocator >, [90](#)
- reduce
 - ceras, [38](#)
- reduce_mean
 - ceras, [39](#)
- reduce_sum
 - ceras, [39](#)
- ReLU
 - ceras, [39](#)
- relu
 - ceras, [39](#)
- remember
 - ceras::session< T, Allocator >, [90](#)
- repeat
 - ceras, [40](#)
- replace_placeholder_with_expression
 - ceras, [40](#)
- repmat
 - ceras, [40](#)
- reset
 - ceras::place_holder< T, Allocator >, [84](#)
 - ceras::tensor< T, Allocator >, [102](#)
 - ceras::variable< T, Allocator >, [114](#)
- reset_action_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [65](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [108](#)
- reset_states
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [63](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [107](#)
 - ceras::variable< T, Allocator >, [114](#)
- Reshape
 - ceras, [40](#)
- reshape
 - ceras, [41](#)
 - ceras::tensor< T, Allocator >, [103](#)
- resize
 - ceras::tensor< T, Allocator >, [103](#)
- restore
 - ceras::session< T, Allocator >, [90](#)
- rho_
 - ceras::adadelta< Loss, T >, [57](#)
 - ceras::rmsprop< Loss, T >, [87](#)
- rhs_input_data_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [65](#)
- rhs_op_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [65](#)

- rms_prop
 - ceras, [18](#)
- RMSprop
 - ceras, [51](#)
- rmsprop
 - ceras::rmsprop< Loss, T >, [86](#)
- row_
 - ceras::view_2d< T >, [118](#)
 - ceras::view_3d< T >, [120](#)
 - ceras::view_4d< T >, [124](#)
- run
 - ceras::session< Tzor >, [90](#)
- save
 - ceras::session< Tzor >, [91](#)
- save_tensor
 - ceras, [41](#)
- self_type
 - ceras::tensor< T, Allocator >, [96](#)
- selu
 - ceras, [41](#)
- serialize
 - ceras::session< Tzor >, [91](#)
- session
 - ceras::session< Tzor >, [89](#)
- SGD
 - ceras, [51](#)
- sgd
 - ceras::sgd< Loss, T >, [93](#)
- shape
 - ceras::constant< Tzor >, [71](#)
 - ceras::tensor< T, Allocator >, [103](#)
 - ceras::variable< Tzor >, [114](#)
 - ceras::view_2d< T >, [118](#)
- shape_
 - ceras::tensor< T, Allocator >, [104](#)
- shape_hint_
 - ceras::place_holder_state< Tzor >, [85](#)
- shared_vector
 - ceras::tensor< T, Allocator >, [96](#)
- shrink_to
 - ceras::tensor< T, Allocator >, [103](#)
- sigmoid
 - ceras, [41](#)
- size
 - ceras::tensor< T, Allocator >, [103](#)
 - ceras::view_2d< T >, [118](#)
- slice
 - ceras::tensor< T, Allocator >, [103](#)
- Softmax
 - ceras, [41](#)
- softmax
 - ceras, [42](#)
- softplus
 - ceras, [42](#)
- softsign
 - ceras, [42](#)
- square
 - ceras, [42](#)
- squared_loss
 - ceras, [42](#)
- squeeze
 - ceras, [43](#)
- standard_deviation
 - ceras, [43](#)
- state_
 - ceras::variable< Tzor >, [114](#)
- stateful_
 - ceras::variable< Tzor >, [115](#)
- STB_IMAGE_IMPLEMENTATION
 - includes.hpp, [128](#)
- STB_IMAGE_RESIZE_IMPLEMENTATION
 - includes.hpp, [129](#)
- STB_IMAGE_WRITE_IMPLEMENTATION
 - includes.hpp, [129](#)
- Subtract
 - ceras, [43](#)
- sum
 - ceras, [43](#)
- sum_reduce
 - ceras, [44](#)
- tanh
 - ceras, [44](#)
- tap
 - ceras::session< Tzor >, [91](#)
- Tensor
 - ceras, [51](#)
- tensor
 - ceras::tensor< T, Allocator >, [96](#), [97](#)
- tensor_type
 - ceras::adadelta< Loss, T >, [55](#)
 - ceras::adagrad< Loss, T >, [57](#)
 - ceras::adam< Loss, T >, [60](#)
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [62](#)
 - ceras::gradient_descent< Loss, T >, [72](#)
 - ceras::place_holder< Tzor >, [82](#)
 - ceras::rmsprop< Loss, T >, [86](#)
 - ceras::sgd< Loss, T >, [92](#)
 - ceras::tensor_deduction< L, R >, [105](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [108](#)
 - ceras::variable< Tzor >, [111](#)
- tesseract
 - ceras, [18](#)
- train_on_batch
 - ceras::compiled_model< Model, Optimizer, Loss >, [68](#)
- trainable_
 - ceras::variable< Tzor >, [115](#)
- transpose
 - ceras, [44](#)
- transposed_
 - ceras::view_2d< T >, [119](#)
- truncated_normal
 - ceras, [44](#)

Unary_Operator ceras, 45
 ceras, 52
unary_operator
 ceras::unary_operator< Operator, Forward_Action,
 Backward_Action >, 106
up_sampling_2d
 ceras, 44
update_cuda_gemm_threshold
 ceras, 44

Value
 ceras, 52
value
 ceras::value< T >, 109
value_type
 ceras::tensor< T, Allocator >, 96
 ceras::value< T >, 109
Variable
 ceras, 52
variable
 ceras::variable< Tsor >, 112
variable_state_type
 ceras::session< Tsor >, 88
variable_type
 ceras::session< Tsor >, 88
variables_
 ceras::session< Tsor >, 91
variance
 ceras, 45
vector_
 ceras::tensor< T, Allocator >, 104
vector_type
 ceras::tensor< T, Allocator >, 96
version
 ceras, 52
view_2d
 ceras::view_2d< T >, 116, 117
view_3d
 ceras::view_3d< T >, 119
view_4d
 ceras::view_4d< T >, 121
visible_device
 ceras, 52

wrapped_lhs_operator_type
 ceras::binary_operator< Lhs_Operator, Rhc_Operator,
 Forward_Action, Backward_Action >, 62
wrapped_operator_type
 ceras::unary_operator< Operator, Forward_Action,
 Backward_Action >, 106
wrapped_rhs_operator_type
 ceras::binary_operator< Lhs_Operator, Rhc_Operator,
 Forward_Action, Backward_Action >, 62
write_tensor
 ceras, 45

zeros
 ceras, 45
zeros_like