

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	16
5.1.1.1 ada_delta	16
5.1.1.2 ada_grad	16
5.1.1.3 cube	17
5.1.1.4 default_allocator	17
5.1.1.5 matrix	17
5.1.1.6 rms_prop	17
5.1.1.7 tesseract	17
5.1.2 Function Documentation	17
5.1.2.1 abs() [1/2]	17
5.1.2.2 abs() [2/2]	18
5.1.2.3 abs_loss()	18
5.1.2.4 add()	18
5.1.2.5 amax()	18
5.1.2.6 amin()	18
5.1.2.7 as_tensor()	18
5.1.2.8 average_pooling_2d()	19
5.1.2.9 batch_normalization()	19
5.1.2.10 BatchNormalization()	19
5.1.2.11 clip() [1/2]	19
5.1.2.12 clip() [2/2]	19
5.1.2.13 concat() [1/2]	19
5.1.2.14 concat() [2/2]	20
5.1.2.15 concatenate() [1/3]	20
5.1.2.16 concatenate() [2/3]	20
5.1.2.17 concatenate() [3/3]	20
5.1.2.18 Conv2D()	20
5.1.2.19 conv2d()	21
5.1.2.20 copy() [1/2]	21
5.1.2.21 copy() [2/2]	21

5.1.2.22 cross_entropy()	21
5.1.2.23 cross_entropy_loss()	21
5.1.2.24 deep_copy()	22
5.1.2.25 Dense()	22
5.1.2.26 drop_out()	22
5.1.2.27 elementwise_divide()	22
5.1.2.28 elementwise_product() [1/2]	22
5.1.2.29 elementwise_product() [2/2]	22
5.1.2.30 elu()	23
5.1.2.31 empty()	23
5.1.2.32 exp()	23
5.1.2.33 exponential()	23
5.1.2.34 flatten()	23
5.1.2.35 gelu()	23
5.1.2.36 gemm() [1/2]	24
5.1.2.37 gemm() [2/2]	24
5.1.2.38 gemm_cpu()	24
5.1.2.39 get_default_session()	24
5.1.2.40 glorot_uniform()	24
5.1.2.41 hadamard_product() [1/2]	25
5.1.2.42 hadamard_product() [2/2]	25
5.1.2.43 hard_sigmoid()	25
5.1.2.44 has_inf()	25
5.1.2.45 has_nan()	25
5.1.2.46 hinge_loss()	25
5.1.2.47 identity()	26
5.1.2.48 img2col()	26
5.1.2.49 Input()	26
5.1.2.50 instance_normalization()	26
5.1.2.51 is_valid()	26
5.1.2.52 leaky_relu()	26
5.1.2.53 linspace()	27
5.1.2.54 load_tensor()	27
5.1.2.55 log()	27
5.1.2.56 mae()	27
5.1.2.57 max() [1/2]	27
5.1.2.58 max() [2/2]	28
5.1.2.59 max_pooling_2d()	28
5.1.2.60 maximum()	28
5.1.2.61 mean() [1/2]	28
5.1.2.62 mean() [2/2]	28
5.1.2.63 mean_absolute_error()	28

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

5.1.2.106 random_like()	36
5.1.2.107 random_normal_like()	36
5.1.2.108 read_tensor()	36
5.1.2.109 reduce()	37
5.1.2.110 reduce_mean() [1/2]	37
5.1.2.111 reduce_mean() [2/2]	37
5.1.2.112 reduce_sum() [1/2]	37
5.1.2.113 reduce_sum() [2/2]	37
5.1.2.114 relu()	37
5.1.2.115 repeat()	38
5.1.2.116 replace_placeholder_with_expression()	38
5.1.2.117 repmat()	38
5.1.2.118 reshape() [1/2]	38
5.1.2.119 reshape() [2/2]	39
5.1.2.120 save_tensor()	39
5.1.2.121 selu()	39
5.1.2.122 sigmoid()	39
5.1.2.123 softmax() [1/2]	39
5.1.2.124 softmax() [2/2]	39
5.1.2.125 softplus()	40
5.1.2.126 softsign()	40
5.1.2.127 square()	40
5.1.2.128 squared_loss()	40
5.1.2.129 squeeze()	40
5.1.2.130 standard_deviation()	40
5.1.2.131 sum() [1/2]	41
5.1.2.132 sum() [2/2]	41
5.1.2.133 sum_reduce()	41
5.1.2.134 tanh()	41
5.1.2.135 transpose()	41
5.1.2.136 truncated_normal()	41
5.1.2.137 up_sampling_2d()	42
5.1.2.138 update_cuda_gemm_threshold()	42
5.1.2.139 variance()	42
5.1.2.140 write_tensor()	42
5.1.2.141 zeros()	42
5.1.2.142 zeros_like()	42
5.1.3 Variable Documentation	43
5.1.3.1 __version__	43
5.1.3.2 Binary_Operator	43
5.1.3.3 blas_mode	43
5.1.3.4 Constant	43

5.1.3.5 cuda_gemm_threshold	43
5.1.3.6 cuda_mode	43
5.1.3.7 debug_mode	44
5.1.3.8 eps	44
5.1.3.9 Expression	44
5.1.3.10 is_binary_operator_v	44
5.1.3.11 is_constant_v	44
5.1.3.12 is_place_holder_v	44
5.1.3.13 is_tensor_v	44
5.1.3.14 is_unary_operator_v	45
5.1.3.15 is_value_v	45
5.1.3.16 is_variable_v	45
5.1.3.17 is_windows_platform	45
5.1.3.18 learning_phase	45
5.1.3.19 lstm	45
5.1.3.20 make_binary_operator	46
5.1.3.21 make_unary_operator	46
5.1.3.22 Operator	46
5.1.3.23 Place_Holder	46
5.1.3.24 random_generator	46
5.1.3.25 random_seed	47
5.1.3.26 Tensor	47
5.1.3.27 Unary_Operator	47
5.1.3.28 Value	47
5.1.3.29 Variable	47
5.1.3.30 version	47
5.1.3.31 visible_device	47
6 Class Documentation	49
6.1 ceras::adadelat< Loss, T > Struct Template Reference	49
6.1.1 Member Typedef Documentation	49
6.1.1.1 tensor_type	50
6.1.2 Constructor & Destructor Documentation	50
6.1.2.1 adadelat()	50
6.1.3 Member Function Documentation	50
6.1.3.1 forward()	50
6.1.4 Member Data Documentation	50
6.1.4.1 iterations_	50
6.1.4.2 learning_rate_	50
6.1.4.3 loss_	51
6.1.4.4 rho_	51
6.2 ceras::adagrad< Loss, T > Struct Template Reference	51

6.2.1 Member Typedef Documentation	51
6.2.1.1 tensor_type	52
6.2.2 Constructor & Destructor Documentation	52
6.2.2.1 adagrad()	52
6.2.3 Member Function Documentation	52
6.2.3.1 forward()	52
6.2.4 Member Data Documentation	52
6.2.4.1 decay_	52
6.2.4.2 iterations_	52
6.2.4.3 learning_rate_	53
6.2.4.4 loss_	53
6.3 ceras::adam< Loss, T > Struct Template Reference	53
6.3.1 Member Typedef Documentation	54
6.3.1.1 tensor_type	54
6.3.2 Constructor & Destructor Documentation	54
6.3.2.1 adam()	54
6.3.3 Member Function Documentation	54
6.3.3.1 forward()	54
6.3.4 Member Data Documentation	54
6.3.4.1 amsgrad_	54
6.3.4.2 beta_1_	55
6.3.4.3 beta_2_	55
6.3.4.4 iterations_	55
6.3.4.5 learning_rate_	55
6.3.4.6 loss_	55
6.4 ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > Struct Template Reference	55
6.4.1 Member Typedef Documentation	56
6.4.1.1 tensor_type	56
6.4.1.2 wrapped_lhs_operator_type	56
6.4.1.3 wrapped_rhs_operator_type	57
6.4.2 Constructor & Destructor Documentation	57
6.4.2.1 binary_operator()	57
6.4.3 Member Function Documentation	57
6.4.3.1 backward()	57
6.4.3.2 forward()	57
6.4.3.3 reset_states()	58
6.4.4 Member Data Documentation	58
6.4.4.1 backward_action_	58
6.4.4.2 forward_action_	58
6.4.4.3 lhs_input_data_	58
6.4.4.4 lhs_op_	58

6.4.4.5 output_data_	59
6.4.4.6 reset_action_	59
6.4.4.7 rhs_input_data_	59
6.4.4.8 rhs_op_	59
6.5 ceras::constant< Tsor > Struct Template Reference	59
6.5.1 Constructor & Destructor Documentation	60
6.5.1.1 constant()	60
6.5.2 Member Function Documentation	60
6.5.2.1 backward()	60
6.5.2.2 forward()	60
6.5.2.3 shape()	60
6.5.3 Member Data Documentation	60
6.5.3.1 data_	61
6.6 ceras::gradient_descent< Loss, T > Struct Template Reference	61
6.6.1 Member Typedef Documentation	61
6.6.1.1 tensor_type	61
6.6.2 Constructor & Destructor Documentation	62
6.6.2.1 gradient_descent()	62
6.6.3 Member Function Documentation	62
6.6.3.1 forward()	62
6.6.4 Member Data Documentation	62
6.6.4.1 learning_rate_	62
6.6.4.2 loss_	62
6.6.4.3 momentum_	62
6.7 ceras::is_binary_operator< T > Struct Template Reference	63
6.8 ceras::is_binary_operator< binary_operator< Lhs_Operator, Rh_Operator, Forward_Action, Backward_Action > > Struct Template Reference	63
6.9 ceras::is_constant< T > Struct Template Reference	63
6.10 ceras::is_constant< constant< Tsor > > Struct Template Reference	64
6.11 ceras::is_place_holder< T > Struct Template Reference	64
6.12 ceras::is_place_holder< place_holder< Tsor > > Struct Template Reference	64
6.13 ceras::is_tensor< T > Struct Template Reference	65
6.14 ceras::is_tensor< tensor< T, A > > Struct Template Reference	65
6.15 ceras::is_unary_operator< T > Struct Template Reference	65
6.16 ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > > Struct Template Reference	66
6.17 ceras::is_value< T > Struct Template Reference	66
6.18 ceras::is_value< value< T > > Struct Template Reference	66
6.19 ceras::is_variable< T > Struct Template Reference	67
6.20 ceras::is_variable< variable< Tsor > > Struct Template Reference	67
6.21 ceras::model< Ex, Ph > Struct Template Reference	67
6.21.1 Detailed Description	68
6.21.2 Constructor & Destructor Documentation	68

6.21.2.1 model()	68
6.21.3 Member Function Documentation	68
6.21.3.1 operator>()	69
6.21.3.2 predict()	69
6.21.4 Member Data Documentation	70
6.21.4.1 expression_	70
6.21.4.2 place_holder_	70
6.22 ceras::place_holder< Tsr > Struct Template Reference	70
6.22.1 Member Typedef Documentation	71
6.22.1.1 tensor_type	71
6.22.2 Constructor & Destructor Documentation	71
6.22.2.1 place_holder() [1/4]	71
6.22.2.2 place_holder() [2/4]	71
6.22.2.3 place_holder() [3/4]	71
6.22.2.4 place_holder() [4/4]	71
6.22.3 Member Function Documentation	71
6.22.3.1 backward()	72
6.22.3.2 bind()	72
6.22.3.3 forward()	72
6.22.3.4 operator=() [1/2]	72
6.22.3.5 operator=() [2/2]	72
6.22.3.6 reset()	72
6.23 ceras::place_holder_state< Tsr > Struct Template Reference	73
6.23.1 Member Data Documentation	73
6.23.1.1 data_	73
6.23.1.2 shape_hint_	73
6.24 ceras::rmsprop< Loss, T > Struct Template Reference	73
6.24.1 Member Typedef Documentation	74
6.24.1.1 tensor_type	74
6.24.2 Constructor & Destructor Documentation	74
6.24.2.1 rmsprop()	74
6.24.3 Member Function Documentation	74
6.24.3.1 forward()	74
6.24.4 Member Data Documentation	75
6.24.4.1 decay_	75
6.24.4.2 iterations_	75
6.24.4.3 learning_rate_	75
6.24.4.4 loss_	75
6.24.4.5 rho_	75
6.25 ceras::session< Tsr > Struct Template Reference	75
6.25.1 Member Typedef Documentation	76
6.25.1.1 place_holder_type	76

6.25.1.2 <code>variable_state_type</code>	76
6.25.1.3 <code>variable_type</code>	77
6.25.2 Constructor & Destructor Documentation	77
6.25.2.1 <code>session()</code> [1/3]	77
6.25.2.2 <code>session()</code> [2/3]	77
6.25.2.3 <code>session()</code> [3/3]	77
6.25.2.4 <code>~session()</code>	77
6.25.3 Member Function Documentation	77
6.25.3.1 <code>bind()</code>	78
6.25.3.2 <code>deserialize()</code>	78
6.25.3.3 <code>operator=()</code> [1/2]	78
6.25.3.4 <code>operator=()</code> [2/2]	78
6.25.3.5 <code>rebind()</code>	78
6.25.3.6 <code>remember()</code>	78
6.25.3.7 <code>restore()</code>	79
6.25.3.8 <code>run()</code>	79
6.25.3.9 <code>save()</code>	79
6.25.3.10 <code>serialize()</code>	79
6.25.3.11 <code>tap()</code>	79
6.25.4 Member Data Documentation	79
6.25.4.1 <code>place_holders_</code>	79
6.25.4.2 <code>variables_</code>	80
6.26 <code>ceras::sgd< Loss, T ></code> Struct Template Reference	80
6.26.1 Member Typedef Documentation	80
6.26.1.1 <code>tensor_type</code>	81
6.26.2 Constructor & Destructor Documentation	81
6.26.2.1 <code>sgd()</code>	81
6.26.3 Member Function Documentation	81
6.26.3.1 <code>forward()</code>	81
6.26.4 Member Data Documentation	81
6.26.4.1 <code>decay_</code>	81
6.26.4.2 <code>iterations_</code>	81
6.26.4.3 <code>learning_rate_</code>	82
6.26.4.4 <code>loss_</code>	82
6.26.4.5 <code>momentum_</code>	82
6.26.4.6 <code>nesterov_</code>	82
6.27 <code>ceras::tensor< T, Allocator ></code> Struct Template Reference	82
6.27.1 Member Typedef Documentation	84
6.27.1.1 <code>allocator</code>	84
6.27.1.2 <code>self_type</code>	84
6.27.1.3 <code>shared_vector</code>	84
6.27.1.4 <code>value_type</code>	84

6.27.1.5 vector_type	84
6.27.2 Constructor & Destructor Documentation	84
6.27.2.1 tensor() [1/7]	84
6.27.2.2 tensor() [2/7]	85
6.27.2.3 tensor() [3/7]	85
6.27.2.4 tensor() [4/7]	85
6.27.2.5 tensor() [5/7]	85
6.27.2.6 tensor() [6/7]	85
6.27.2.7 tensor() [7/7]	85
6.27.3 Member Function Documentation	86
6.27.3.1 as_scalar()	86
6.27.3.2 begin() [1/2]	86
6.27.3.3 begin() [2/2]	86
6.27.3.4 cbegin()	86
6.27.3.5 cend()	86
6.27.3.6 copy()	86
6.27.3.7 creep_to()	87
6.27.3.8 data() [1/2]	87
6.27.3.9 data() [2/2]	87
6.27.3.10 deep_copy() [1/2]	87
6.27.3.11 deep_copy() [2/2]	87
6.27.3.12 empty()	87
6.27.3.13 end() [1/2]	88
6.27.3.14 end() [2/2]	88
6.27.3.15 map()	88
6.27.3.16 ndim()	88
6.27.3.17 operator*=() [1/2]	88
6.27.3.18 operator*=() [2/2]	88
6.27.3.19 operator+=() [1/2]	89
6.27.3.20 operator+=() [2/2]	89
6.27.3.21 operator-()	89
6.27.3.22 operator-=() [1/2]	89
6.27.3.23 operator-=() [2/2]	89
6.27.3.24 operator/=() [1/2]	89
6.27.3.25 operator/=() [2/2]	90
6.27.3.26 operator=() [1/2]	90
6.27.3.27 operator=() [2/2]	90
6.27.3.28 operator[]() [1/2]	90
6.27.3.29 operator[]() [2/2]	90
6.27.3.30 reset()	90
6.27.3.31 reshape()	91
6.27.3.32 resize()	91

6.27.3.33 shape()	91
6.27.3.34 shrink_to()	91
6.27.3.35 size()	91
6.27.3.36 slice()	91
6.27.4 Member Data Documentation	92
6.27.4.1 memory_offset_	92
6.27.4.2 shape_	92
6.27.4.3 vector_	92
6.28 ceras::tensor_deduction< L, R > Struct Template Reference	92
6.28.1 Member Typedef Documentation	92
6.28.1.1 op_type	92
6.28.1.2 tensor_type	93
6.29 ceras::unary_operator< Operator, Forward_Action, Backward_Action > Struct Template Reference	93
6.29.1 Member Typedef Documentation	93
6.29.1.1 wrapped_operator_type	94
6.29.2 Constructor & Destructor Documentation	94
6.29.2.1 unary_operator()	94
6.29.3 Member Function Documentation	94
6.29.3.1 backward()	94
6.29.3.2 forward()	94
6.29.3.3 reset_states()	94
6.29.4 Member Data Documentation	94
6.29.4.1 backward_action_	95
6.29.4.2 forward_action_	95
6.29.4.3 input_data_	95
6.29.4.4 op_	95
6.29.4.5 output_data_	95
6.29.4.6 reset_action_	95
6.29.4.7 tensor_type	96
6.30 ceras::value< T > Struct Template Reference	96
6.30.1 Member Typedef Documentation	96
6.30.1.1 value_type	96
6.30.2 Constructor & Destructor Documentation	97
6.30.2.1 value() [1/4]	97
6.30.2.2 value() [2/4]	97
6.30.2.3 value() [3/4]	97
6.30.2.4 value() [4/4]	97
6.30.3 Member Function Documentation	97
6.30.3.1 backward()	97
6.30.3.2 forward()	98
6.30.3.3 operator=() [1/2]	98
6.30.3.4 operator=() [2/2]	98

6.30.4 Member Data Documentation	98
6.30.4.1 data_	98
6.31 ceras::variable< Tsor > Struct Template Reference	98
6.31.1 Member Typedef Documentation	99
6.31.1.1 tensor_type	99
6.31.2 Constructor & Destructor Documentation	99
6.31.2.1 variable() [1/4]	99
6.31.2.2 variable() [2/4]	100
6.31.2.3 variable() [3/4]	100
6.31.2.4 variable() [4/4]	100
6.31.3 Member Function Documentation	100
6.31.3.1 backward()	100
6.31.3.2 data() [1/2]	100
6.31.3.3 data() [2/2]	100
6.31.3.4 forward()	101
6.31.3.5 gradient() [1/2]	101
6.31.3.6 gradient() [2/2]	101
6.31.3.7 operator=() [1/2]	101
6.31.3.8 operator=() [2/2]	101
6.31.3.9 reset()	101
6.31.3.10 reset_states()	101
6.31.3.11 shape()	102
6.31.4 Member Data Documentation	102
6.31.4.1 state_	102
6.31.4.2 stateful_	102
6.31.4.3 trainable_	102
6.32 ceras::variable_state< Tsor > Struct Template Reference	102
6.32.1 Member Data Documentation	102
6.32.1.1 data_	103
6.32.1.2 gradient_	103
6.33 ceras::view_2d< T > Struct Template Reference	103
6.33.1 Constructor & Destructor Documentation	103
6.33.1.1 view_2d() [1/3]	104
6.33.1.2 view_2d() [2/3]	104
6.33.1.3 view_2d() [3/3]	104
6.33.2 Member Function Documentation	104
6.33.2.1 data() [1/2]	104
6.33.2.2 data() [2/2]	104
6.33.2.3 operator[]() [1/2]	105
6.33.2.4 operator[]() [2/2]	105
6.33.2.5 shape()	105
6.33.2.6 size()	105

6.33.3 Member Data Documentation	105
6.33.3.1 col_	105
6.33.3.2 data_	105
6.33.3.3 row_	106
6.33.3.4 transposed_	106
6.34 ceras::view_3d< T > Struct Template Reference	106
6.34.1 Constructor & Destructor Documentation	106
6.34.1.1 view_3d()	106
6.34.2 Member Function Documentation	107
6.34.2.1 operator[]() [1/2]	107
6.34.2.2 operator[]() [2/2]	107
6.34.3 Member Data Documentation	107
6.34.3.1 channel_	107
6.34.3.2 col_	107
6.34.3.3 data_	107
6.34.3.4 row_	108
6.35 ceras::view_4d< T > Struct Template Reference	108
6.35.1 Detailed Description	108
6.35.2 Constructor & Destructor Documentation	108
6.35.2.1 view_4d()	109
6.35.3 Member Function Documentation	110
6.35.3.1 operator[]() [1/2]	110
6.35.3.2 operator[]() [2/2]	110
6.35.4 Member Data Documentation	111
6.35.4.1 batch_size_	111
6.35.4.2 channel_	111
6.35.4.3 col_	111
6.35.4.4 data_	111
6.35.4.5 row_	111
7 File Documentation	113
7.1 /data/structured_folders/workspace/github.repo/ceras/include/activation.hpp File Reference	113
7.2 /data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp File Reference	114
7.3 /data/structured_folders/workspace/github.repo/ceras/include/config.hpp File Reference	114
7.4 /data/structured_folders/workspace/github.repo/ceras/include/constant.hpp File Reference	115
7.5 /data/structured_folders/workspace/github.repo/ceras/include/includes.hpp File Reference	115
7.5.1 Macro Definition Documentation	116
7.5.1.1 STB_IMAGE_IMPLEMENTATION	116
7.5.1.2 STB_IMAGE_RESIZE_IMPLEMENTATION	116
7.5.1.3 STB_IMAGE_WRITE_IMPLEMENTATION	116
7.6 /data/structured_folders/workspace/github.repo/ceras/include/keras.hpp File Reference	117
7.7 /data/structured_folders/workspace/github.repo/ceras/include/layer.hpp File Reference	117

7.8 /data/structured_folders/workspace/github.repo/ceras/include/loss.hpp File Reference	117
7.9 /data/structured_folders/workspace/github.repo/ceras/include/model.hpp File Reference	118
7.10 /data/structured_folders/workspace/github.repo/ceras/include/operation.hpp File Reference	119
7.11 /data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp File Reference	121
7.12 /data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp File Reference	122
7.13 /data/structured_folders/workspace/github.repo/ceras/include/recurrent_operation.hpp File Reference	123
7.14 /data/structured_folders/workspace/github.repo/ceras/include/session.hpp File Reference	123
7.15 /data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp File Reference	124
7.16 /data/structured_folders/workspace/github.repo/ceras/include/value.hpp File Reference	127
7.17 /data/structured_folders/workspace/github.repo/ceras/include/variable.hpp File Reference	128

Index	131
--------------	------------

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

ceras	9
---------------------------------	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

ceras::constant< Tzor >	59
enable_id	
ceras::adadelta< Loss, T >	49
ceras::adagrad< Loss, T >	51
ceras::adam< Loss, T >	53
ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >	55
ceras::gradient_descent< Loss, T >	61
ceras::place_holder< Tzor >	70
ceras::rmsprop< Loss, T >	73
ceras::sgd< Loss, T >	80
ceras::tensor< T, Allocator >	82
ceras::unary_operator< Operator, Forward_Action, Backward_Action >	93
ceras::variable< Tzor >	98
enable_shared	
ceras::adadelta< Loss, T >	49
ceras::adagrad< Loss, T >	51
ceras::adam< Loss, T >	53
ceras::gradient_descent< Loss, T >	61
ceras::rmsprop< Loss, T >	73
ceras::sgd< Loss, T >	80
enable_shared_state	
ceras::place_holder< Tzor >	70
std::false_type	
ceras::is_binary_operator< T >	63
ceras::is_constant< T >	63
ceras::is_place_holder< T >	64
ceras::is_tensor< T >	65
ceras::is_unary_operator< T >	65
ceras::is_value< T >	66
ceras::is_variable< T >	67
ceras::model< Ex, Ph >	67
ceras::place_holder_state< Tzor >	73
ceras::session< Tzor >	75
ceras::tensor_deduction< L, R >	92
ceras::tensor_deduction< Lhs_Operator, Rhc_Operator >	92

std::true_type	
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > >	63
ceras::is_constant< constant< Tsor > >	64
ceras::is_place_holder< place_holder< Tsor > >	64
ceras::is_tensor< tensor< T, A > >	65
ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >	66
ceras::is_value< value< T > >	66
ceras::is_variable< variable< Tsor > >	67
ceras::value< T >	96
ceras::variable_state< Tsor >	102
ceras::view_2d< T >	103
ceras::view_3d< T >	106
ceras::view_4d< T >	108

Chapter 3

Class Index

3.1 Class List

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

ceras::adadelta< Loss, T >	49
ceras::adagrad< Loss, T >	51
ceras::adam< Loss, T >	53
ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >	55
ceras::constant< Tsor >	59
ceras::gradient_descent< Loss, T >	61
ceras::is_binary_operator< T >	63
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > >	63
ceras::is_constant< T >	63
ceras::is_constant< constant< Tsor > >	64
ceras::is_place_holder< T >	64
ceras::is_place_holder< place_holder< Tsor > >	64
ceras::is_tensor< T >	65
ceras::is_tensor< tensor< T, A > >	65
ceras::is_unary_operator< T >	65
ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >	66
ceras::is_value< T >	66
ceras::is_value< value< T > >	66
ceras::is_variable< T >	67
ceras::is_variable< variable< Tsor > >	67
ceras::model< Ex, Ph >	67
ceras::place_holder< Tsor >	70
ceras::place_holder_state< Tsor >	73
ceras::rmsprop< Loss, T >	73
ceras::session< Tsor >	75
ceras::sgd< Loss, T >	80
ceras::tensor< T, Allocator >	82
ceras::tensor_deduction< L, R >	92
ceras::unary_operator< Operator, Forward_Action, Backward_Action >	93
ceras::value< T >	96
ceras::variable< Tsor >	98
ceras::variable_state< Tsor >	102
ceras::view_2d< T >	103
ceras::view_3d< T >	106
ceras::view_4d< T >	108

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	113
/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp	114
/data/structured_folders/workspace/github.repo/ceras/include/config.hpp	114
/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp	115
/data/structured_folders/workspace/github.repo/ceras/include/includes.hpp	115
/data/structured_folders/workspace/github.repo/ceras/include/keras.hpp	117
/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp	117
/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp	117
/data/structured_folders/workspace/github.repo/ceras/include/model.hpp	118
/data/structured_folders/workspace/github.repo/ceras/include/operation.hpp	119
/data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp	121
/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp	122
/data/structured_folders/workspace/github.repo/ceras/include/recurrent_operation.hpp	123
/data/structured_folders/workspace/github.repo/ceras/include/session.hpp	123
/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp	124
/data/structured_folders/workspace/github.repo/ceras/include/value.hpp	127
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp	128

Chapter 5

Namespace Documentation

5.1 ceras Namespace Reference

Classes

- struct [constant](#)
- struct [is_constant](#)
- struct [is_constant< constant< Tsor > >](#)
- struct [model](#)
- struct [unary_operator](#)
- struct [binary_operator](#)
- struct [is_unary_operator](#)
- struct [is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >](#)
- struct [is_binary_operator](#)
- struct [is_binary_operator< binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action > >](#)
- struct [sgd](#)
- struct [adagrad](#)
- struct [rmsprop](#)
- struct [adadelat](#)
- struct [adam](#)
- struct [gradient_descent](#)
- struct [place_holder_state](#)
- struct [place_holder](#)
- struct [is_place_holder](#)
- struct [is_place_holder< place_holder< Tsor > >](#)
- struct [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` = `adadelta`< Loss, T >
- `template<typename T >`
using `default_allocator` = `std::allocator`< T >
- `template<typename T >`
using `matrix` = `view_2d`< T >
- `template<typename T >`
using `cube` = `view_3d`< T >
- `template<typename T >`
using `tesseract` = `view_4d`< T >

Functions

- `template<Expression Ex>`
`constexpr auto softmax` (Ex const &ex) noexcept
- `template<Expression Ex>`
`auto selu` (Ex const &ex) noexcept
- `template<Expression Ex>`
`auto softplus` (Ex const &ex) noexcept
- `template<Expression Ex>`
`auto softsign` (Ex const &ex) noexcept
- `template<Expression Ex>`
`auto sigmoid` (Ex const &ex) noexcept
- `template<Expression Ex>`
`auto tanh` (Ex const &ex) noexcept
- `template<Expression Ex>`
`auto relu` (Ex const &ex) noexcept
- `template<typename T >`
requires `std::floating_point`< T > `auto leaky_relu` (T const factor) noexcept
- `template<Expression Ex>`
`auto negative_relu` (Ex const &ex) noexcept
- `template<typename T >`
requires `std::floating_point`< T > `auto elu` (T const alpha) 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)
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto mean_squared_logarithmic_error` (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto squared_loss` (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept

- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto mean_squared_error (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto mse (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto abs_loss (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto mean_absolute_error (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto mae (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto cross_entropy (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto cross_entropy_loss (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto hinge_loss (Lhs_Expression const &lhs_ex, Rh_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<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto plus (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto operator+ (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`auto operator* (Lhs_Expression const &lhs_ex, Rh_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 Rh_Expression>`
`constexpr auto elementwise_product (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto hadamard_product (Lhs_Expression const &lhs_ex, Rh_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 Rh_Expression>`
`constexpr auto minus (Lhs_Expression const &lhs_ex, Rh_Expression const &rhs_ex) noexcept`
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
`constexpr auto operator- (Lhs_Expression const &lhs_ex, Rh_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 Rh_Expression>`
constexpr auto `concatenate` (Lhs_Expression const &`lhs_ex`, Rh_Expression const &`rhs_ex`) noexcept
- auto `concatenate` (unsigned long `axe=-1`)
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
constexpr auto `concat` (Lhs_Expression const &`lhs_ex`, Rh_Expression const &`rhs_ex`) noexcept
- auto `concat` (unsigned long `axe=-1`)
- `template<Expression Lhs_Expression, Expression Rh_Expression>`
constexpr auto `maximum` (Lhs_Expression const &`lhs_ex`, Rh_Expression const &`rhs_ex`) noexcept
- `template<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 Rh_Expression>`
constexpr auto `copy` (Lhs_Expression const &`lhs_ex`, Rh_Expression 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 > glorot_uniform (std::initializer_list< unsigned long > shape)`
- `template<Tensor Tsor>`
`Tsor deep_copy (Tsor const &tsor)`
- `template<Tensor Tsor>`
`Tsor copy (Tsor const &tsor)`
- `template<Tensor Tsor>`
`Tsor concatenate (Tsor const &lhs, Tsor const &rhs, unsigned long axis=0) noexcept`
- `template<Tensor Tsor>`
`Tsor repmat (Tsor const &tsor, unsigned long row_rep, unsigned long col_rep)`
- `template<Tensor Tsor>`
`constexpr bool empty (Tsor const &tsor) noexcept`
- `template<typename T , typename A = default_allocator<T>>>`
`constexpr tensor< T, A > zeros (std::vector< unsigned long > const &shape)`
- `template<Tensor Tsor>`
`constexpr Tsor zeros_like (Tsor const &tsor)`
- `template<typename T , typename A = default_allocator<T>>>`
`constexpr tensor< T, A > ones (std::vector< unsigned long > const &shape)`
- `template<Tensor Tsor>`
`constexpr Tsor ones_like (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto max (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto amax (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto min (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto amin (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto sum (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto mean (Tsor const &tsor)`
- `template<Tensor Tsor>`
`auto norm (Tsor const &tsor)`
- `template<Tensor Tsor>`
`Tsor abs (Tsor const &tsor)`
- `template<Tensor Tsor>`
`Tsor softmax (Tsor const &tsor)`
- `template<Tensor Tsor>`
`bool has_nan (Tsor const &tsor)`
- `template<Tensor Tsor>`
`bool has_inf (Tsor const &tsor)`

- `template<Tensor Tsor>`
`bool is_valid (Tsor const &tsor)`
- `template<Tensor Tsor, typename Function >`
`Tsor reduce (Tsor const &ts, unsigned long axis, typename Tsor::value_type const &init, Function const &func, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`
`Tsor sum (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`
`requires std::floating_point< typename Tsor::value_type > Tsor mean (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`
`requires std::floating_point< typename Tsor::value_type > Tsor variance (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`
`requires std::floating_point< typename Tsor::value_type > Tsor standard_deviation (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept`
- `template<Tensor Tsor>`
`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>`
- `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.
- `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 adadelata< 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()

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

5.1.2.5 amax()

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

5.1.2.6 amin()

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

5.1.2.7 as_tensor()

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

5.1.2.8 average_pooling_2d()

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

5.1.2.9 batch_normalization()

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

5.1.2.10 BatchNormalization()

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

5.1.2.11 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]
```

5.1.2.12 clip() [2/2]

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

5.1.2.13 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]
```

5.1.2.14 concat() [2/2]

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

5.1.2.15 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]
```

5.1.2.16 concatenate() [2/3]

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

5.1.2.17 concatenate() [3/3]

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

5.1.2.18 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]
```

5.1.2.19 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]
```

5.1.2.20 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]
```

5.1.2.21 copy() [2/2]

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

5.1.2.22 cross_entropy()

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

5.1.2.23 cross_entropy_loss()

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

5.1.2.24 deep_copy()

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

5.1.2.25 Dense()

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

5.1.2.26 drop_out()

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

5.1.2.27 elementwise_divide()

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

5.1.2.28 elementwise_product() [1/2]

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

5.1.2.29 elementwise_product() [2/2]

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

5.1.2.30 elu()

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

5.1.2.31 empty()

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

5.1.2.32 exp()

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

5.1.2.33 exponential()

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

5.1.2.34 flatten()

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

5.1.2.35 gelu()

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

5.1.2.36 gemm() [1/2]

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

5.1.2.37 gemm() [2/2]

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

5.1.2.38 gemm_cpu()

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

5.1.2.39 get_default_session()

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

5.1.2.40 glorot_uniform()

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


5.1.2.41 hadamard_product() [1/2]

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

5.1.2.42 hadamard_product() [2/2]

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

5.1.2.43 hard_sigmoid()

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

5.1.2.44 has_inf()

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

5.1.2.45 has_nan()

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

5.1.2.46 hinge_loss()

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

5.1.2.47 identity()

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

5.1.2.48 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]
```

5.1.2.49 Input()

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

5.1.2.50 instance_normalization()

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

5.1.2.51 is_valid()

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

5.1.2.52 leaky_relu()

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

5.1.2.53 linspace()

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

5.1.2.54 load_tensor()

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

5.1.2.55 log()

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

5.1.2.56 mae()

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

5.1.2.57 max() [1/2]

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

5.1.2.58 max() [2/2]

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

5.1.2.59 max_pooling_2d()

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

5.1.2.60 maximum()

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

5.1.2.61 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]
```

5.1.2.62 mean() [2/2]

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

5.1.2.63 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]
```

5.1.2.64 mean_reduce()

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

5.1.2.65 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]
```

5.1.2.66 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]
```

5.1.2.67 min() [1/2]

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

5.1.2.68 min() [2/2]

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

5.1.2.69 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]
```

5.1.2.70 minus() [2/2]

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

5.1.2.71 mse()

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

5.1.2.72 multiply() [1/2]

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

5.1.2.73 multiply() [2/2]

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

5.1.2.74 negative()

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

5.1.2.75 negative_relu()

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

5.1.2.76 norm()

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

5.1.2.77 normalization_batch()

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

5.1.2.78 normalization_instance()

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

5.1.2.79 ones()

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

5.1.2.80 ones_like()

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

5.1.2.81 operator"!="()

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

5.1.2.82 operator*() [1/4]

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

5.1.2.83 operator*() [2/4]

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

5.1.2.84 operator*() [3/4]

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

5.1.2.85 operator*() [4/4]

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

5.1.2.86 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]
```

5.1.2.87 operator+() [2/4]

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


5.1.2.88 operator+() [3/4]

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

5.1.2.89 operator+() [4/4]

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

5.1.2.90 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]
```

5.1.2.91 operator-() [2/4]

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

5.1.2.92 operator-() [3/4]

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

5.1.2.93 operator-() [4/4]

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

5.1.2.94 operator/()

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

5.1.2.95 operator<()

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

5.1.2.96 operator<<()

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

5.1.2.97 operator<=()

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

5.1.2.98 operator==() [1/2]

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

5.1.2.99 operator==() [2/2]

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

5.1.2.100 operator>()

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

5.1.2.101 operator>=()

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

5.1.2.102 plus()

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

5.1.2.103 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} )
```

5.1.2.104 randn_like()

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

5.1.2.105 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} )
```

5.1.2.106 random_like()

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

5.1.2.107 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)
```

5.1.2.108 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 )
```

5.1.2.109 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]
```

5.1.2.110 reduce_mean() [1/2]

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

5.1.2.111 reduce_mean() [2/2]

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

5.1.2.112 reduce_sum() [1/2]

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

5.1.2.113 reduce_sum() [2/2]

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

5.1.2.114 relu()

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

5.1.2.115 repeat()

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

5.1.2.116 replace_placeholder_with_expression()

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

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

Parameters

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

Returns

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

5.1.2.117 repmat()

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

5.1.2.118 reshape() [1/2]

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

5.1.2.119 reshape() [2/2]

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

5.1.2.120 save_tensor()

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

5.1.2.121 selu()

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

5.1.2.122 sigmoid()

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

5.1.2.123 softmax() [1/2]

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

5.1.2.124 softmax() [2/2]

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

5.1.2.125 `softplus()`

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

5.1.2.126 `softsign()`

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

5.1.2.127 `square()`

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

5.1.2.128 `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]
```

5.1.2.129 `squeeze()`

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

5.1.2.130 `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]
```


5.1.2.131 sum() [1/2]

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

5.1.2.132 sum() [2/2]

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

5.1.2.133 sum_reduce()

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

5.1.2.134 tanh()

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

5.1.2.135 transpose()

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

5.1.2.136 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} )
```

5.1.2.137 up_sampling_2d()

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

5.1.2.138 update_cuda_gemm_threshold()

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

5.1.2.139 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]
```

5.1.2.140 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 )
```

5.1.2.141 zeros()

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

5.1.2.142 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 `Binary_Operator`

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

A type that represents a binary operator.

```
@concept Binary_Operator<>
```

5.1.3.3 `blas_mode`

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

5.1.3.4 `Constant`

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

5.1.3.5 `cuda_gemm_threshold`

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

5.1.3.6 `cuda_mode`

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

5.1.3.7 debug_mode

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

5.1.3.8 eps

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

5.1.3.9 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<>
```

5.1.3.10 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.

5.1.3.11 is_constant_v

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

5.1.3.12 is_place_holder_v

```
template<class T >
constexpr bool ceras::is_place_holder_v = is\_place\_holder<T>::value [inline], [constexpr]
```

5.1.3.13 is_tensor_v

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

5.1.3.14 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`.

5.1.3.15 is_value_v

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

5.1.3.16 is_variable_v

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

5.1.3.17 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();
}
```

5.1.3.18 learning_phase

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

5.1.3.19 lstm

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

5.1.3.20 make_binary_operator

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

Initial value:

```
= [] ( auto const& binary_forward_action, auto const& binary_backward_action, std::string const&
    name="Anonymous Binary Operator", 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;
    };
};
```

5.1.3.21 make_unary_operator

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

Initial value:

```
= [] ( auto const& unary_forward_action, auto const& unary_backward_action, std::string const&
    name="Anonymous Unary Operator", 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;
    };
};
```

5.1.3.22 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<>

5.1.3.23 Place_Holder

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

5.1.3.24 random_generator

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

5.1.3.25 random_seed

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

5.1.3.26 Tensor

```
template<typename T >  
concept ceras::Tensor = is_tensor_v<T>
```

5.1.3.27 Unary_Operator

```
template<typename T >  
concept ceras::Unary_Operator = is_unary_operator_v<T>
```

A type that represents an unary operator.

```
@concept Unary_Operator<>
```

5.1.3.28 Value

```
template<typename T >  
concept ceras::Value = is_value_v<T>
```

5.1.3.29 Variable

```
template<typename T >  
concept ceras::Variable = is_variable_v<T>
```

5.1.3.30 version

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

5.1.3.31 visible_device

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

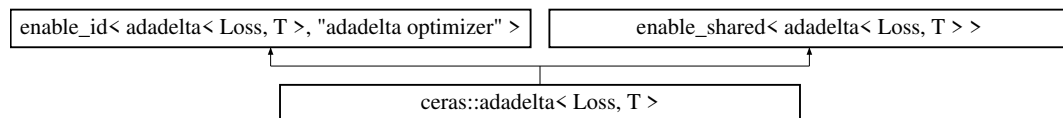

Chapter 6

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_
```

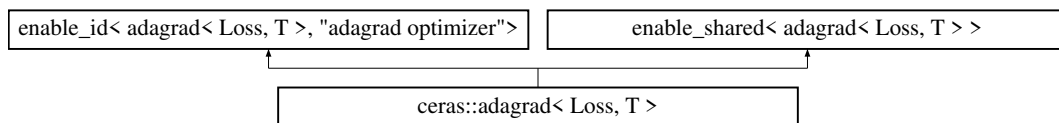
The documentation for this struct was generated from the following file:

- /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_
```

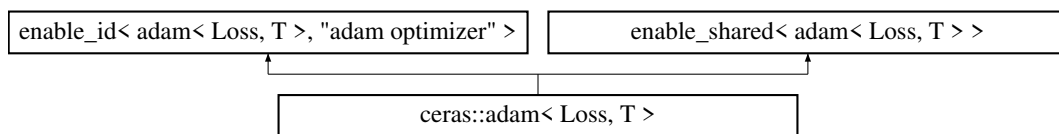
The documentation for this struct was generated from the following file:

- /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_
```

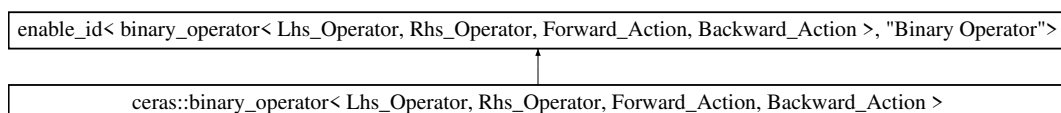
The documentation for this struct was generated from the following file:

- /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::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

- T_{sor} [data_](#)

6.5.1 Constructor & Destructor Documentation

6.5.1.1 constant()

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

6.5.2 Member Function Documentation

6.5.2.1 backward()

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

6.5.2.2 forward()

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

6.5.2.3 shape()

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

6.5.3 Member Data Documentation

6.5.3.1 data_

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

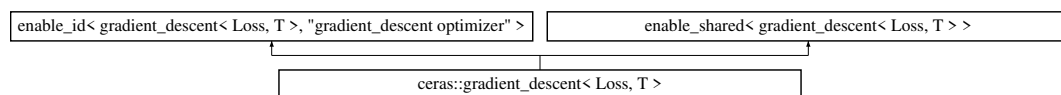
The documentation for this struct was generated from the following file:

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

6.6 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_](#)

6.6.1 Member Typedef Documentation

6.6.1.1 tensor_type

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

6.6.2 Constructor & Destructor Documentation

6.6.2.1 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]
```

6.6.3 Member Function Documentation

6.6.3.1 forward()

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

6.6.4 Member Data Documentation

6.6.4.1 learning_rate_

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

6.6.4.2 loss_

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

6.6.4.3 momentum_

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

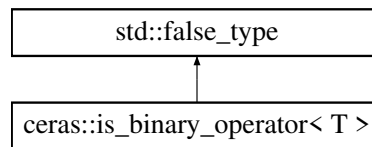
The documentation for this struct was generated from the following file:

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

6.7 ceras::is_binary_operator< T > Struct Template Reference

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

Inheritance diagram for ceras::is_binary_operator< T >:



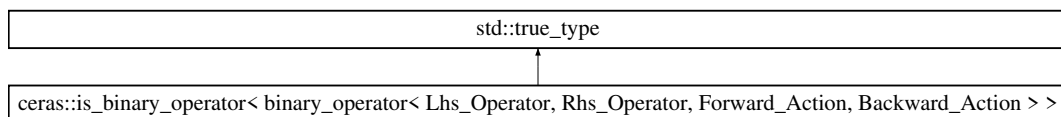
The documentation for this struct was generated from the following file:

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

6.8 ceras::is_binary_operator< binary_operator< Lhs_Operator, Rh_Operator, Forward_Action, Backward_Action > > Struct Template Reference

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

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



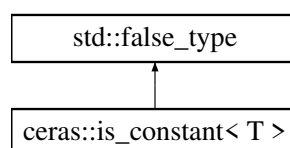
The documentation for this struct was generated from the following file:

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

6.9 ceras::is_constant< T > Struct Template Reference

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

Inheritance diagram for ceras::is_constant< T >:



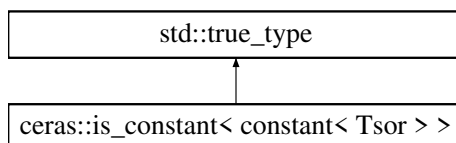
The documentation for this struct was generated from the following file:

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

6.10 `ceras::is_constant< constant< Tsor > >` Struct Template Reference

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

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



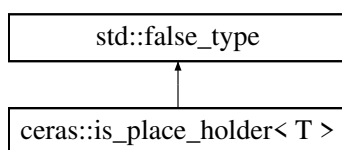
The documentation for this struct was generated from the following file:

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

6.11 `ceras::is_place_holder< T >` Struct Template Reference

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

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



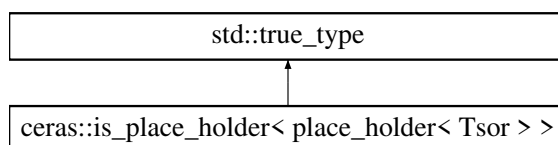
The documentation for this struct was generated from the following file:

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

6.12 `ceras::is_place_holder< place_holder< Tsor > >` Struct Template Reference

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

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



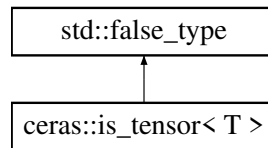
The documentation for this struct was generated from the following file:

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

6.13 ceras::is_tensor< T > Struct Template Reference

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

Inheritance diagram for ceras::is_tensor< T >:



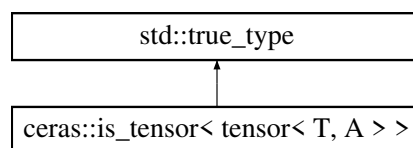
The documentation for this struct was generated from the following file:

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

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

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

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



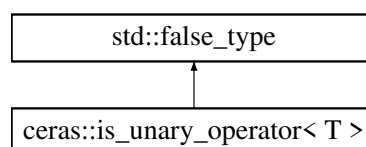
The documentation for this struct was generated from the following file:

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

6.15 ceras::is_unary_operator< T > Struct Template Reference

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

Inheritance diagram for ceras::is_unary_operator< T >:



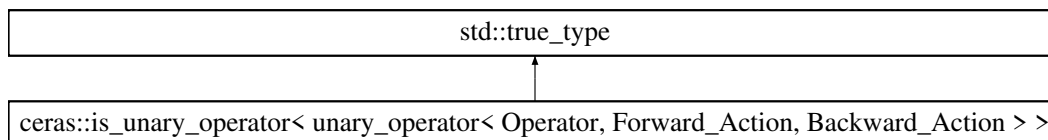
The documentation for this struct was generated from the following file:

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

6.16 `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 > >`:



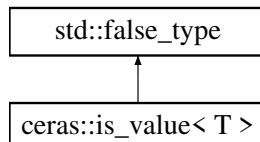
The documentation for this struct was generated from the following file:

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

6.17 `ceras::is_value< T >` Struct Template Reference

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

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



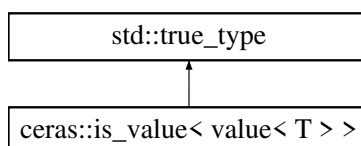
The documentation for this struct was generated from the following file:

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

6.18 `ceras::is_value< value< T > >` Struct Template Reference

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

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



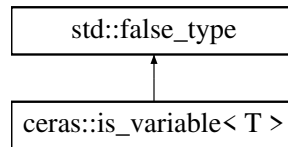
The documentation for this struct was generated from the following file:

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

6.19 ceras::is_variable< T > Struct Template Reference

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

Inheritance diagram for ceras::is_variable< T >:



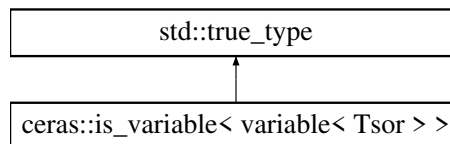
The documentation for this struct was generated from the following file:

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

6.20 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

6.21 ceras::model< Ex, Ph > Struct Template Reference

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

Public Member Functions

- [model](#) (Ph const &[place_holder](#), Ex const &expression)
- `template<Tensor Tsor>`
`auto predict (Tsor const &input_tensor)`
- `template<Expression Exp>`
`auto operator\(\) (Exp const &ex) const noexcept`

Public Attributes

- Ex [expression_](#)
output layer of the model.
- Ph [place_holder_](#)

6.21.1 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

6.21.2 Constructor & Destructor Documentation

6.21.2.1 model()

```
template<Expression Ex, Place_Holder Ph>
ceras::model< Ex, Ph >::model (
    Ph const & place_holder,
    Ex 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 };
```

6.21.3 Member Function Documentation

6.21.3.1 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
```

6.21.3.2 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)
```

6.21.4 Member Data Documentation

6.21.4.1 expression_

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

output layer of the model.

6.21.4.2 place_holder_

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

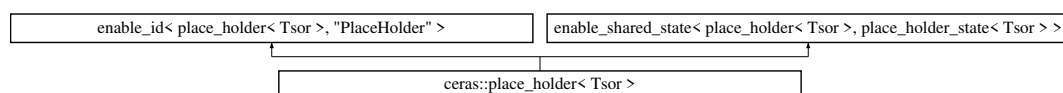
The documentation for this struct was generated from the following file:

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

6.22 [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

6.22.1 Member Typedef Documentation

6.22.1.1 tensor_type

```
template<Tensor Tzor>
typedef Tzor ceras::place_holder< Tzor >::tensor_type
```

6.22.2 Constructor & Destructor Documentation

6.22.2.1 place_holder() [1/4]

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

6.22.2.2 place_holder() [2/4]

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

6.22.2.3 place_holder() [3/4]

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

6.22.2.4 place_holder() [4/4]

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

6.22.3 Member Function Documentation

6.22.3.1 backward()

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

6.22.3.2 bind()

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

6.22.3.3 forward()

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

6.22.3.4 operator=() [1/2]

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

6.22.3.5 operator=() [2/2]

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

6.22.3.6 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](#)

6.23 ceras::place_holder_state< Tzor > Struct Template Reference

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

Public Attributes

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

6.23.1 Member Data Documentation

6.23.1.1 data_

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

6.23.1.2 shape_hint_

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

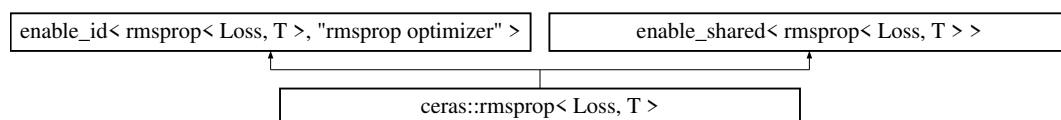
The documentation for this struct was generated from the following file:

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

6.24 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_](#)

6.24.1 Member Typedef Documentation

6.24.1.1 tensor_type

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

6.24.2 Constructor & Destructor Documentation

6.24.2.1 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]
```

6.24.3 Member Function Documentation

6.24.3.1 forward()

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

6.24.4 Member Data Documentation

6.24.4.1 decay_

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

6.24.4.2 iterations_

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

6.24.4.3 learning_rate_

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

6.24.4.4 loss_

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

6.24.4.5 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](#)

6.25 ceras::session< Tsr > Struct Template Reference

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

Public Types

- typedef [place_holder](#)< Tsor > [place_holder_type](#)
- typedef [variable](#)< Tsor > [variable_type](#)
- typedef [variable_state](#)< Tsor > [variable_state_type](#)

Public Member Functions

- [session](#) ()
- [session](#) ([session](#) const &)=delete
- [session](#) ([session](#) &&)=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_](#)

6.25.1 Member Typedef Documentation

6.25.1.1 place_holder_type

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

6.25.1.2 variable_state_type

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

6.25.1.3 variable_type

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

6.25.2 Constructor & Destructor Documentation

6.25.2.1 session() [1/3]

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

6.25.2.2 session() [2/3]

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

6.25.2.3 session() [3/3]

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

6.25.2.4 ~session()

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

6.25.3 Member Function Documentation

6.25.3.1 bind()

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

6.25.3.2 deserialize()

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

6.25.3.3 operator=() [1/2]

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

6.25.3.4 operator=() [2/2]

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

6.25.3.5 rebind()

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

6.25.3.6 remember()

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

6.25.3.7 restore()

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

6.25.3.8 run()

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

6.25.3.9 save()

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

6.25.3.10 serialize()

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

6.25.3.11 tap()

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

6.25.4 Member Data Documentation

6.25.4.1 place_holders_

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

6.25.4.2 variables_

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

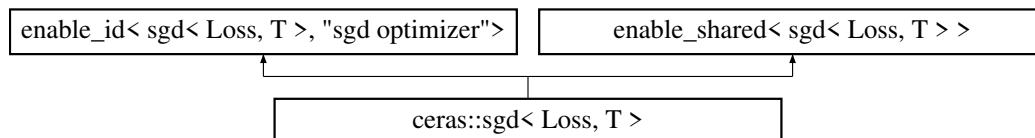
The documentation for this struct was generated from the following file:

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

6.26 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_`

6.26.1 Member Typedef Documentation

6.26.1.1 tensor_type

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

6.26.2 Constructor & Destructor Documentation

6.26.2.1 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]
```

6.26.3 Member Function Documentation

6.26.3.1 forward()

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

6.26.4 Member Data Documentation

6.26.4.1 decay_

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

6.26.4.2 iterations_

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

6.26.4.3 learning_rate_

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

6.26.4.4 loss_

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

6.26.4.5 momentum_

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

6.26.4.6 nesterov_

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

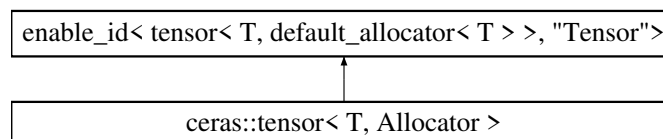
The documentation for this struct was generated from the following file:

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

6.27 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

Public Attributes

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

6.27.1 Member Typedef Documentation

6.27.1.1 allocator

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

6.27.1.2 self_type

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

6.27.1.3 shared_vector

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

6.27.1.4 value_type

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

6.27.1.5 vector_type

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

6.27.2 Constructor & Destructor Documentation

6.27.2.1 tensor() [1/7]

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

6.27.2.2 `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]
```

6.27.2.3 `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]
```

6.27.2.4 `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]
```

6.27.2.5 `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]
```

6.27.2.6 `tensor()` [6/7]

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

6.27.2.7 `tensor()` [7/7]

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

6.27.3 Member Function Documentation

6.27.3.1 as_scalar()

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

6.27.3.2 begin() [1/2]

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

6.27.3.3 begin() [2/2]

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

6.27.3.4 cbegin()

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

6.27.3.5 cend()

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

6.27.3.6 copy()

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

6.27.3.7 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]
```

6.27.3.8 data() [1/2]

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

6.27.3.9 data() [2/2]

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

6.27.3.10 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]
```

6.27.3.11 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]
```

6.27.3.12 empty()

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

6.27.3.13 end() [1/2]

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

6.27.3.14 end() [2/2]

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

6.27.3.15 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]
```

6.27.3.16 ndim()

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

6.27.3.17 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]
```

6.27.3.18 operator*=() [2/2]

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


6.27.3.19 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]
```

6.27.3.20 operator+=() [2/2]

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

6.27.3.21 operator-()

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

6.27.3.22 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]
```

6.27.3.23 operator-=() [2/2]

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

6.27.3.24 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]
```

6.27.3.25 operator/=() [2/2]

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

6.27.3.26 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]
```

6.27.3.27 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]
```

6.27.3.28 operator[]() [1/2]

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

6.27.3.29 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]
```

6.27.3.30 reset()

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

6.27.3.31 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]
```

6.27.3.32 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]
```

6.27.3.33 shape()

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

6.27.3.34 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]
```

6.27.3.35 size()

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

6.27.3.36 slice()

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

6.27.4 Member Data Documentation

6.27.4.1 memory_offset_

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

6.27.4.2 shape_

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

6.27.4.3 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

6.28 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())>

6.28.1 Member Typedef Documentation

6.28.1.1 op_type

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

6.28.1.2 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
```

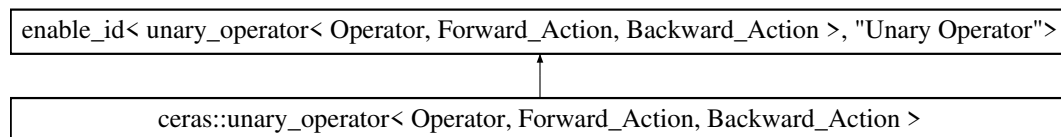
The documentation for this struct was generated from the following file:

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

6.29 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_

6.29.1 Member Typedef Documentation

6.29.1.1 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
```

6.29.2 Constructor & Destructor Documentation

6.29.2.1 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]
```

6.29.3 Member Function Documentation

6.29.3.1 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]
```

6.29.3.2 forward()

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

6.29.3.3 reset_states()

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
void ceras::unary\_operator< Operator, Forward_Action, Backward_Action >::reset\_states ( )
[inline]
```

6.29.4 Member Data Documentation

6.29.4.1 backward_action_

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

6.29.4.2 forward_action_

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

6.29.4.3 input_data_

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

6.29.4.4 op_

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

6.29.4.5 output_data_

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

6.29.4.6 reset_action_

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

6.29.4.7 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](#)

6.30 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_](#)

6.30.1 Member Typedef Documentation

6.30.1.1 value_type

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


6.30.2 Constructor & Destructor Documentation

6.30.2.1 value() [1/4]

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

6.30.2.2 value() [2/4]

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

6.30.2.3 value() [3/4]

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

6.30.2.4 value() [4/4]

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

6.30.3 Member Function Documentation

6.30.3.1 backward()

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

6.30.3.2 forward()

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

6.30.3.3 operator=() [1/2]

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

6.30.3.4 operator=() [2/2]

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

6.30.4 Member Data Documentation

6.30.4.1 data_

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

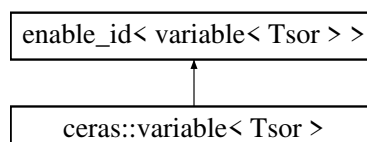
The documentation for this struct was generated from the following file:

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

6.31 ceras::variable< Tsor > Struct Template Reference

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

Inheritance diagram for ceras::variable< Tsor >:



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
- 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_](#)

6.31.1 Member Typedef Documentation

6.31.1.1 tensor_type

```
template<Tensor Tzor>
typedef Tzor ceras::variable< Tzor >::tensor\_type
```

6.31.2 Constructor & Destructor Documentation

6.31.2.1 variable() [1/4]

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

6.31.2.2 variable() [2/4]

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

6.31.2.3 variable() [3/4]

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

6.31.2.4 variable() [4/4]

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

6.31.3 Member Function Documentation**6.31.3.1 backward()**

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

6.31.3.2 data() [1/2]

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

6.31.3.3 data() [2/2]

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

6.31.3.4 forward()

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

6.31.3.5 gradient() [1/2]

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

6.31.3.6 gradient() [2/2]

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

6.31.3.7 operator=() [1/2]

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

6.31.3.8 operator=() [2/2]

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

6.31.3.9 reset()

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

6.31.3.10 reset_states()

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

6.31.3.11 shape()

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

6.31.4 Member Data Documentation

6.31.4.1 state_

```
template<Tensor Tsor>
std::shared_ptr<variable\_state<Tsor> > ceras::variable< Tsor >::state_
```

6.31.4.2 stateful_

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

6.31.4.3 trainable_

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

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

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

6.32 [ceras::variable_state](#)< Tsor > Struct Template Reference

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

Public Attributes

- Tsor [data_](#)
- Tsor [gradient_](#)

6.32.1 Member Data Documentation

6.32.1.1 data_

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

6.32.1.2 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](#)

6.33 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_](#)

6.33.1 Constructor & Destructor Documentation

6.33.1.1 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]
```

6.33.1.2 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]
```

6.33.1.3 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]
```

6.33.2 Member Function Documentation**6.33.2.1 data() [1/2]**

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

6.33.2.2 data() [2/2]

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


6.33.2.3 operator[]() [1/2]

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

6.33.2.4 operator[]() [2/2]

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

6.33.2.5 shape()

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

6.33.2.6 size()

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

6.33.3 Member Data Documentation

6.33.3.1 col_

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

6.33.3.2 data_

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

6.33.3.3 row_

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

6.33.3.4 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

6.34 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_](#)

6.34.1 Constructor & Destructor Documentation

6.34.1.1 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]
```

6.34.2 Member Function Documentation

6.34.2.1 operator[]() [1/2]

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

6.34.2.2 operator[]() [2/2]

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

6.34.3 Member Data Documentation

6.34.3.1 channel_

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

6.34.3.2 col_

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

6.34.3.3 data_

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

6.34.3.4 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

6.35 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.

6.35.1 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].

6.35.2 Constructor & Destructor Documentation

6.35.2.1 `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.

6.35.3 Member Function Documentation

6.35.3.1 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];
```

6.35.3.2 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;
```

6.35.4 Member Data Documentation

6.35.4.1 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.

6.35.4.2 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.

6.35.4.3 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.

6.35.4.4 data_

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

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

6.35.4.5 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 >`
`requires std::floating_point< T > auto ceras::elu (T const alpha) 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"
```

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/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](#)

7.5.1 Macro Definition Documentation

7.5.1.1 STB_IMAGE_IMPLEMENTATION

```
#define STB_IMAGE_IMPLEMENTATION
```

7.5.1.2 STB_IMAGE_RESIZE_IMPLEMENTATION

```
#define STB_IMAGE_RESIZE_IMPLEMENTATION
```

7.5.1.3 STB_IMAGE_WRITE_IMPLEMENTATION

```
#define STB_IMAGE_WRITE_IMPLEMENTATION
```

7.6 /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"
```

7.7 /data/structured_↵/folders/workspace/github.repo/ceras/include/layer.hpp File Reference

```
#include "../operation.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)

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

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

Namespaces

- [ceras](#)

Functions

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

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

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

Classes

- `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`)

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

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

Classes

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

Namespaces

- `ceras`

Functions

- `template<Expression 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 Rhx_Expression>
constexpr auto [ceras::concat](#) (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_ex) noexcept
- auto [ceras::concat](#) (unsigned long axe=-1)
- template<Expression Lhs_Expression, Expression Rhx_Expression>
constexpr auto [ceras::maximum](#) (Lhs_Expression const &lhx_ex, Rhx_Expression const &rhx_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.

7.11 /data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp File Reference

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

Classes

- struct [ceras::sgd](#)< Loss, T >
- struct [ceras::adagrad](#)< Loss, T >
- struct [ceras::rmsprop](#)< Loss, T >
- struct [ceras::adadelta](#)< Loss, T >
- struct [ceras::adam](#)< Loss, T >
- struct [ceras::gradient_descent](#)< Loss, T >

Namespaces

- [ceras](#)

Typedefs

- `template<typename Loss , typename T >`
using [ceras::ada_grad](#) = `adagrad< Loss, T >`
- `template<typename Loss , typename T >`
using [ceras::rms_prop](#) = `rmsprop< Loss, T >`
- `template<typename Loss , typename T >`
using [ceras::ada_delta](#) = `adadelta< Loss, T >`

7.12 `/data/structured_` `folders/workspace/github.repo/ceras/include/place_holder.hpp` File Reference

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

Classes

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

Namespaces

- [ceras](#)

Functions

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

Variables

- template<class T >
constexpr bool [ceras::is_place_holder_v](#) = is_place_holder<T>::value
- template<typename T >
concept [ceras::Place_Holder](#) = is_place_holder_v<T>

7.13 /data/structured_folders/workspace/github.repo/ceras/include/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](#)

7.14 /data/structured_folders/workspace/github.repo/ceras/include/session.hpp File Reference

```
#include "../includes.hpp"
#include "../tensor.hpp"
#include "../place_holder.hpp"
#include "../variable.hpp"
#include "../utils/singleton.hpp"
#include "../utils/debug.hpp"
```

Classes

- struct [ceras::session](#)< Tsor >

Namespaces

- [ceras](#)

Functions

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

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

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

Classes

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

Namespaces

- [ceras](#)

Typedefs

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

Functions

- `template<typename T , typename A = default_allocator<T>>>`
`constexpr tensor< T, A > ceras::as_tensor (T val) noexcept`
- `template<Tensor Tsor, typename CharT , typename Traits >`
`std::basic_ostream< CharT, Traits > & ceras::operator<< (std::basic_ostream< CharT, Traits > &os_, Tsor const &tsor)`
- `template<typename T >`
`requires std::floating_point< T > void ceras::gemm_cpu (T const *A, bool a_transposed, T const *B, bool b_transposed, unsigned long m, unsigned long n, unsigned long k, T *C)`
- `void ceras::update_cuda_gemm_threshold ()`
- `template<typename T >`
`requires std::floating_point< T > void ceras::gemm (T const *A, bool a_transposed, T const *B, bool b_transposed, unsigned long m, unsigned long n, unsigned long k, T *C)`
- `template<typename T >`
`requires std::floating_point< T > void ceras::gemm (view_2d< T > const &x, view_2d< T > const &y, view_2d< T > &ans)`
- `template<Tensor Tsor>`
`Tsor ceras::add (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator+ (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator+ (typename Tsor::value_type const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator+ (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::minus (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator- (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator- (typename Tsor::value_type const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator- (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator* (typename Tsor::value_type const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator* (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator/ (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::reshape (Tsor const &ts, std::vector< unsigned long > const &new_shape)`
- `template<Tensor Tsor>`
`void ceras::multiply (Tsor const &lhs, Tsor const &rhs, Tsor &ans) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::multiply (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::operator* (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::elementwise_product (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::hadamard_product (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::elementwise_divide (Tsor const &lhs, Tsor const &rhs) noexcept`
- `template<Tensor Tsor>`
`Tsor ceras::repeat (Tsor const &tsor, unsigned long n)`
- `template<Tensor Tsor>`
`Tsor ceras::reduce_sum (Tsor const &tsor)`

- `template<Tensor Tsor>`
`Tsor ceras::reduce_mean (Tsor const &tsor)`
- `template<Tensor Tsor>`
`Tsor ceras::clip (Tsor &tsor, typename Tsor::value_type lower=0, typename Tsor::value_type upper=1)`
- `template<Tensor Tsor>`
`Tsor ceras::squeeze (Tsor const &tsor)`
- `template<typename T, typename A = default_allocator<T>>>`
`tensor< T, A > ceras::randn (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1})`
- `template<typename T, typename A = default_allocator<T>>>`
`tensor< T, A > ceras::truncated_normal (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1}, T lower=T{0}, T upper=T{1})`
- `template<typename T, typename A = default_allocator<T>>>`
`tensor< T, A > ceras::random (std::vector< unsigned long > const &shape, T min=T{0}, T max=T{1})`
- `template<Tensor 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>

7.16 /data/structured_folders/workspace/github.repo/ceras/include/value.hpp File Reference

```
#include "../includes.hpp"
#include "../tensor.hpp"
```

```
#include "../utils/id.hpp"
#include "../utils/better_assert.hpp"
#include "../utils/enable_shared.hpp"
```

Classes

- struct [ceras::value< T >](#)
- struct [ceras::is_value< T >](#)
- struct [ceras::is_value< value< T > >](#)
- struct [ceras::tensor_deduction< L, R >](#)

Namespaces

- [ceras](#)

Variables

- template<class T >
constexpr bool [ceras::is_value_v](#) = is_value<T>::value
- template<typename T >
concept [ceras::Value](#) = is_value_v<T>

7.17 /data/structured_↵ folders/workspace/github.repo/ceras/include/variable.hpp File Reference

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

Classes

- struct [ceras::variable_state< Ttor >](#)
- struct [ceras::variable< Ttor >](#)
- struct [ceras::is_variable< T >](#)
- struct [ceras::is_variable< variable< Ttor > >](#)

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](#), 113
[/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp](#), 114
[/data/structured_folders/workspace/github.repo/ceras/include/config.hpp](#), 114
[/data/structured_folders/workspace/github.repo/ceras/include/ceras_tensor.hpp](#), 115
[/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp](#), 115
[/data/structured_folders/workspace/github.repo/ceras/include/keras.hpp](#), 117
[/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp](#), 117
[/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp](#), 117
[/data/structured_folders/workspace/github.repo/ceras/include/model.hpp](#), 118
[/data/structured_folders/workspace/github.repo/ceras/include/operator.hpp](#), 119
[/data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp](#), 121
[/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp](#), 122
[/data/structured_folders/workspace/github.repo/ceras/include/recurrent_operation.hpp](#), 123
[/data/structured_folders/workspace/github.repo/ceras/include/session.hpp](#), 123
[/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp](#), 124
[/data/structured_folders/workspace/github.repo/ceras/include/value.hpp](#), 127
[/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp](#), 128
__version__
 ceras, 43
~session
 ceras::session< Tsor >, 77
abs
 ceras, 17
abs_loss
 ceras, 18
ada_delta
 ceras, 16
ada_grad
 ceras, 16
adadelat
 ceras::adadelat< Loss, T >, 50
adagrad
 ceras::adagrad< Loss, T >, 52
adam
 ceras::adam< Loss, T >, 54
add
 ceras, 18
allocator
 ceras::tensor< T, Allocator >, 84
amax
 ceras, 19
amin
 ceras, 18
amsgrad
 ceras::amsgrad< Loss, T >, 54
as_scalar
 ceras::tensor< T, Allocator >, 86
as_tensor
 ceras, 18
average_pooling_2d
 ceras, 19
backward
 ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >, 57
 ceras::constant< Tsor >, 60
 ceras::place_holder< Tsor >, 71
 ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 94
 ceras::value< T >, 97
 ceras::variable< Tsor >, 100
backward_action_
 ceras::binary_operator< Lhs_Operator, Rhc_Operator, Forward_Action, Backward_Action >, 58
 ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 94
batch_normalization
 ceras, 19
batch_size_
 ceras::view_4d< T >, 111
BatchNormalization
 ceras, 19
begin
 ceras::tensor< T, Allocator >, 86
beta_1_
 ceras::adam< Loss, T >, 54
beta_2_
 ceras::adam< Loss, T >, 55
Binary_Operator
 ceras, 43
binary_operator

- ceras::binary_operator< Lhs_Operator, Rh_Operator, Forward_Action, Backward_Action >, 57
- bind
 - ceras::place_holder< Tsor >, 72
 - ceras::session< Tsor >, 77
- blas_mode
 - ceras, 43
- cbegin
 - ceras::tensor< T, Allocator >, 86
- cend
 - ceras::tensor< T, Allocator >, 86
- ceras, 9
 - __version__, 43
 - abs, 17
 - abs_loss, 18
 - ada_delta, 16
 - ada_grad, 16
 - add, 18
 - amax, 18
 - amin, 18
 - as_tensor, 18
 - average_pooling_2d, 18
 - batch_normalization, 19
 - BatchNormalization, 19
 - Binary_Operator, 43
 - blas_mode, 43
 - clip, 19
 - concat, 19
 - concatenate, 20
 - Constant, 43
 - Conv2D, 20
 - conv2d, 20
 - copy, 21
 - cross_entropy, 21
 - cross_entropy_loss, 21
 - cube, 16
 - cuda_gemm_threshold, 43
 - cuda_mode, 43
 - debug_mode, 43
 - deep_copy, 21
 - default_allocator, 17
 - Dense, 22
 - drop_out, 22
 - elementwise_divide, 22
 - elementwise_product, 22
 - elu, 22
 - empty, 23
 - eps, 44
 - exp, 23
 - exponential, 23
 - Expression, 44
 - flatten, 23
 - gelu, 23
 - gemm, 23, 24
 - gemm_cpu, 24
 - get_default_session, 24
 - glorot_uniform, 24
 - hadamard_product, 24, 25
 - hard_sigmoid, 25
 - has_inf, 25
 - has_nan, 25
 - hinge_loss, 25
 - identity, 25
 - img2col, 26
 - Input, 26
 - instance_normalization, 26
 - is_binary_operator_v, 44
 - is_constant_v, 44
 - is_place_holder_v, 44
 - is_tensor_v, 44
 - is_unary_operator_v, 44
 - is_valid, 26
 - is_value_v, 45
 - is_variable_v, 45
 - is_windows_platform, 45
 - leaky_relu, 26
 - learning_phase, 45
 - linspace, 26
 - load_tensor, 27
 - log, 27
 - lstm, 45
 - mae, 27
 - make_binary_operator, 45
 - make_unary_operator, 46
 - matrix, 17
 - max, 27
 - max_pooling_2d, 28
 - maximum, 28
 - mean, 28
 - mean_absolute_error, 28
 - mean_reduce, 28
 - mean_squared_error, 29
 - mean_squared_logarithmic_error, 29
 - min, 29
 - minus, 29
 - mse, 30
 - multiply, 30
 - negative, 30
 - negative_relu, 30
 - norm, 30
 - normalization_batch, 31
 - normalization_instance, 31
 - ones, 31
 - ones_like, 31
 - Operator, 46
 - operator!=, 31
 - operator<, 34
 - operator<<, 34
 - operator<=, 34
 - operator>, 34
 - operator>=, 35
 - operator*, 31, 32
 - operator+, 32, 33
 - operator-, 33
 - operator/, 33
 - operator==, 34

- Place_Holder, 46
- plus, 35
- randn, 35
- randn_like, 35
- random, 35
- random_generator, 46
- random_like, 36
- random_normal_like, 36
- random_seed, 46
- read_tensor, 36
- reduce, 36
- reduce_mean, 37
- reduce_sum, 37
- relu, 37
- repeat, 37
- replace_placeholder_with_expression, 38
- repmat, 38
- reshape, 38
- rms_prop, 17
- save_tensor, 39
- selu, 39
- sigmoid, 39
- softmax, 39
- softplus, 39
- softsign, 40
- square, 40
- squared_loss, 40
- squeeze, 40
- standard_deviation, 40
- sum, 40, 41
- sum_reduce, 41
- tanh, 41
- Tensor, 47
- tesseract, 17
- transpose, 41
- truncated_normal, 41
- Unary_Operator, 47
- up_sampling_2d, 41
- update_cuda_gemm_threshold, 42
- Value, 47
- Variable, 47
- variance, 42
- version, 47
- visible_device, 47
- write_tensor, 42
- zeros, 42
- zeros_like, 42
- ceras::adadelta< Loss, T >, 49
 - adadelta, 50
 - forward, 50
 - iterations_, 50
 - learning_rate_, 50
 - loss_, 50
 - rho_, 51
 - tensor_type, 49
- ceras::adagrad< Loss, T >, 51
 - adagrad, 52
 - decay_, 52
 - forward, 52
 - iterations_, 52
 - learning_rate_, 52
 - loss_, 53
 - tensor_type, 51
- ceras::adam< Loss, T >, 53
 - adam, 54
 - amsgrad_, 54
 - beta_1_, 54
 - beta_2_, 55
 - forward, 54
 - iterations_, 55
 - learning_rate_, 55
 - loss_, 55
 - tensor_type, 54
- ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 55
 - backward, 57
 - backward_action_, 58
 - binary_operator, 57
 - forward, 57
 - forward_action_, 58
 - lhs_input_data_, 58
 - lhs_op_, 58
 - output_data_, 58
 - reset_action_, 59
 - reset_states, 57
 - rhs_input_data_, 59
 - rhs_op_, 59
 - tensor_type, 56
 - wrapped_lhs_operator_type, 56
 - wrapped_rhs_operator_type, 56
- ceras::constant< T >, 59
 - backward, 60
 - constant, 60
 - data_, 60
 - forward, 60
 - shape, 60
- ceras::gradient_descent< Loss, T >, 61
 - forward, 62
 - gradient_descent, 62
 - learning_rate_, 62
 - loss_, 62
 - momentum_, 62
 - tensor_type, 61
- ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > >, 63
- ceras::is_binary_operator< T >, 63
- ceras::is_constant< constant< T > >, 64
- ceras::is_constant< T >, 63
- ceras::is_place_holder< place_holder< T > >, 64
- ceras::is_place_holder< T >, 64
- ceras::is_tensor< T >, 65
- ceras::is_tensor< tensor< T, A > >, 65
- ceras::is_unary_operator< T >, 65
- ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >, 66

- ceras::is_value< T >, 66
- ceras::is_value< value< T > >, 66
- ceras::is_variable< T >, 67
- ceras::is_variable< variable< Tzor > >, 67
- ceras::model< Ex, Ph >, 67
 - expression_, 70
 - model, 68
 - operator(), 68
 - place_holder_, 70
 - predict, 69
- ceras::place_holder< Tzor >, 70
 - backward, 71
 - bind, 72
 - forward, 72
 - operator=, 72
 - place_holder, 71
 - reset, 72
 - tensor_type, 71
- ceras::place_holder_state< Tzor >, 73
 - data_, 73
 - shape_hint_, 73
- ceras::rmsprop< Loss, T >, 73
 - decay_, 75
 - forward, 74
 - iterations_, 75
 - learning_rate_, 75
 - loss_, 75
 - rho_, 75
 - rmsprop, 74
 - tensor_type, 74
- ceras::session< Tzor >, 75
 - ~session, 77
 - bind, 77
 - deserialize, 78
 - operator=, 78
 - place_holder_type, 76
 - place_holders_, 79
 - rebind, 78
 - remember, 78
 - restore, 78
 - run, 79
 - save, 79
 - serialize, 79
 - session, 77
 - tap, 79
 - variable_state_type, 76
 - variable_type, 76
 - variables_, 79
- ceras::sgd< Loss, T >, 80
 - decay_, 81
 - forward, 81
 - iterations_, 81
 - learning_rate_, 81
 - loss_, 82
 - momentum_, 82
 - nesterov_, 82
 - sgd, 81
 - tensor_type, 80
- ceras::tensor< T, Allocator >, 82
 - allocator, 84
 - as_scalar, 86
 - begin, 86
 - cbegin, 86
 - cend, 86
 - copy, 86
 - creep_to, 86
 - data, 87
 - deep_copy, 87
 - empty, 87
 - end, 87, 88
 - map, 88
 - memory_offset_, 92
 - ndim, 88
 - operator*=, 88
 - operator+=, 88, 89
 - operator-, 89
 - operator-=, 89
 - operator/=: 89
 - operator=, 90
 - operator[], 90
 - reset, 90
 - reshape, 90
 - resize, 91
 - self_type, 84
 - shape, 91
 - shape_, 92
 - shared_vector, 84
 - shrink_to, 91
 - size, 91
 - slice, 91
 - tensor, 84, 85
 - value_type, 84
 - vector_, 92
 - vector_type, 84
- ceras::tensor_deduction< L, R >, 92
 - op_type, 92
 - tensor_type, 92
- ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 93
 - backward, 94
 - backward_action_, 94
 - forward, 94
 - forward_action_, 95
 - input_data_, 95
 - op_, 95
 - output_data_, 95
 - reset_action_, 95
 - reset_states, 94
 - tensor_type, 95
 - unary_operator, 94
 - wrapped_operator_type, 93
- ceras::value< T >, 96
 - backward, 97
 - data_, 98
 - forward, 97
 - operator=, 98

- value, [97](#)
- value_type, [96](#)
- ceras::variable< Tzor >, [98](#)
 - backward, [100](#)
 - data, [100](#)
 - forward, [100](#)
 - gradient, [101](#)
 - operator=, [101](#)
 - reset, [101](#)
 - reset_states, [101](#)
 - shape, [101](#)
 - state_, [102](#)
 - stateful_, [102](#)
 - tensor_type, [99](#)
 - trainable_, [102](#)
 - variable, [99](#), [100](#)
- ceras::variable_state< Tzor >, [102](#)
 - data_, [102](#)
 - gradient_, [103](#)
- ceras::view_2d< T >, [103](#)
 - col_, [105](#)
 - data, [104](#)
 - data_, [105](#)
 - operator[], [104](#), [105](#)
 - row_, [105](#)
 - shape, [105](#)
 - size, [105](#)
 - transposed_, [106](#)
 - view_2d, [103](#), [104](#)
- ceras::view_3d< T >, [106](#)
 - channel_, [107](#)
 - col_, [107](#)
 - data_, [107](#)
 - operator[], [107](#)
 - row_, [107](#)
 - view_3d, [106](#)
- ceras::view_4d< T >, [108](#)
 - batch_size_, [111](#)
 - channel_, [111](#)
 - col_, [111](#)
 - data_, [111](#)
 - operator[], [110](#)
 - row_, [111](#)
 - view_4d, [108](#)
- channel_
 - ceras::view_3d< T >, [107](#)
 - ceras::view_4d< T >, [111](#)
- clip
 - ceras, [19](#)
- col_
 - ceras::view_2d< T >, [105](#)
 - ceras::view_3d< T >, [107](#)
 - ceras::view_4d< T >, [111](#)
- concat
 - ceras, [19](#)
- concatenate
 - ceras, [20](#)
- Constant
 - ceras, [43](#)
- constant
 - ceras::constant< Tzor >, [60](#)
- Conv2D
 - ceras, [20](#)
- conv2d
 - ceras, [20](#)
- copy
 - ceras, [21](#)
 - ceras::tensor< T, Allocator >, [86](#)
- creep_to
 - ceras::tensor< T, Allocator >, [86](#)
- cross_entropy
 - ceras, [21](#)
- cross_entropy_loss
 - ceras, [21](#)
- cube
 - ceras, [16](#)
- cuda_gemm_threshold
 - ceras, [43](#)
- cuda_mode
 - ceras, [43](#)
- data
 - ceras::tensor< T, Allocator >, [87](#)
 - ceras::variable< Tzor >, [100](#)
 - ceras::view_2d< T >, [104](#)
- data_
 - ceras::constant< Tzor >, [60](#)
 - ceras::place_holder_state< Tzor >, [73](#)
 - ceras::value< T >, [98](#)
 - ceras::variable_state< Tzor >, [102](#)
 - ceras::view_2d< T >, [105](#)
 - ceras::view_3d< T >, [107](#)
 - ceras::view_4d< T >, [111](#)
- debug_mode
 - ceras, [43](#)
- decay_
 - ceras::adagrad< Loss, T >, [52](#)
 - ceras::rmsprop< Loss, T >, [75](#)
 - ceras::sgd< Loss, T >, [81](#)
- deep_copy
 - ceras, [21](#)
 - ceras::tensor< T, Allocator >, [87](#)
- default_allocator
 - ceras, [17](#)
- Dense
 - ceras, [22](#)
- deserialize
 - ceras::session< Tzor >, [78](#)
- drop_out
 - ceras, [22](#)
- elementwise_divide
 - ceras, [22](#)
- elementwise_product
 - ceras, [22](#)
- elu
 - ceras, [22](#)

- empty
 - ceras, [23](#)
 - ceras::tensor< T, Allocator >, [87](#)
- end
 - ceras::tensor< T, Allocator >, [87](#), [88](#)
- eps
 - ceras, [44](#)
- exp
 - ceras, [23](#)
- exponential
 - ceras, [23](#)
- Expression
 - ceras, [44](#)
- expression_
 - ceras::model< Ex, Ph >, [70](#)
- flatten
 - ceras, [23](#)
- forward
 - ceras::adadelta< Loss, T >, [50](#)
 - ceras::adagrad< Loss, T >, [52](#)
 - ceras::adam< Loss, T >, [54](#)
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [57](#)
 - ceras::constant< Tzor >, [60](#)
 - ceras::gradient_descent< Loss, T >, [62](#)
 - ceras::place_holder< Tzor >, [72](#)
 - ceras::rmsprop< Loss, T >, [74](#)
 - ceras::sgd< Loss, T >, [81](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [94](#)
 - ceras::value< T >, [97](#)
 - ceras::variable< Tzor >, [100](#)
- forward_action_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [58](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [95](#)
- gelu
 - ceras, [23](#)
- gemm
 - ceras, [23](#), [24](#)
- gemm_cpu
 - ceras, [24](#)
- get_default_session
 - ceras, [24](#)
- glorot_uniform
 - ceras, [24](#)
- gradient
 - ceras::variable< Tzor >, [101](#)
- gradient_
 - ceras::variable_state< Tzor >, [103](#)
- gradient_descent
 - ceras::gradient_descent< Loss, T >, [62](#)
- hadamard_product
 - ceras, [24](#), [25](#)
- hard_sigmoid
 - ceras, [25](#)
- has_inf
 - ceras, [25](#)
- has_nan
 - ceras, [25](#)
- hinge_loss
 - ceras, [25](#)
- identity
 - ceras, [25](#)
- img2col
 - ceras, [26](#)
- includes.hpp
 - STB_IMAGE_IMPLEMENTATION, [116](#)
 - STB_IMAGE_RESIZE_IMPLEMENTATION, [116](#)
 - STB_IMAGE_WRITE_IMPLEMENTATION, [116](#)
- Input
 - ceras, [26](#)
- input_data_
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [95](#)
- instance_normalization
 - ceras, [26](#)
- is_binary_operator_v
 - ceras, [44](#)
- is_constant_v
 - ceras, [44](#)
- is_place_holder_v
 - ceras, [44](#)
- is_tensor_v
 - ceras, [44](#)
- is_unary_operator_v
 - ceras, [44](#)
- is_valid
 - ceras, [26](#)
- is_value_v
 - ceras, [45](#)
- is_variable_v
 - ceras, [45](#)
- is_windows_platform
 - ceras, [45](#)
- iterations_
 - ceras::adadelta< Loss, T >, [50](#)
 - ceras::adagrad< Loss, T >, [52](#)
 - ceras::adam< Loss, T >, [55](#)
 - ceras::rmsprop< Loss, T >, [75](#)
 - ceras::sgd< Loss, T >, [81](#)
- leaky_relu
 - ceras, [26](#)
- learning_phase
 - ceras, [45](#)
- learning_rate_
 - ceras::adadelta< Loss, T >, [50](#)
 - ceras::adagrad< Loss, T >, [52](#)
 - ceras::adam< Loss, T >, [55](#)
 - ceras::gradient_descent< Loss, T >, [62](#)
 - ceras::rmsprop< Loss, T >, [75](#)
 - ceras::sgd< Loss, T >, [81](#)

- lhs_input_data_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 58
- lhs_op_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 58
- linspace
 - ceras, 26
- load_tensor
 - ceras, 27
- log
 - ceras, 27
- loss_
 - ceras::adadelta< Loss, T >, 50
 - ceras::adagrad< Loss, T >, 53
 - ceras::adam< Loss, T >, 55
 - ceras::gradient_descent< Loss, T >, 62
 - ceras::rmsprop< Loss, T >, 75
 - ceras::sgd< Loss, T >, 82
- lstm
 - ceras, 45
- mae
 - ceras, 27
- make_binary_operator
 - ceras, 45
- make_unary_operator
 - ceras, 46
- map
 - ceras::tensor< T, Allocator >, 88
- matrix
 - ceras, 17
- max
 - ceras, 27
- max_pooling_2d
 - ceras, 28
- maximum
 - ceras, 28
- mean
 - ceras, 28
- mean_absolute_error
 - ceras, 28
- mean_reduce
 - ceras, 28
- mean_squared_error
 - ceras, 29
- mean_squared_logarithmic_error
 - ceras, 29
- memory_offset_
 - ceras::tensor< T, Allocator >, 92
- min
 - ceras, 29
- minus
 - ceras, 29
- model
 - ceras::model< Ex, Ph >, 68
- momentum_
 - ceras::gradient_descent< Loss, T >, 62
 - ceras::sgd< Loss, T >, 82
- mse
 - ceras, 30
- multiply
 - ceras, 30
- ndim
 - ceras::tensor< T, Allocator >, 88
- negative
 - ceras, 30
- negative_relu
 - ceras, 30
- nesterov_
 - ceras::sgd< Loss, T >, 82
- norm
 - ceras, 30
- normalization_batch
 - ceras, 31
- normalization_instance
 - ceras, 31
- ones
 - ceras, 31
- ones_like
 - ceras, 31
- op_
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 95
- op_type
 - ceras::tensor_deduction< L, R >, 92
- Operator
 - ceras, 46
- operator!=
 - ceras, 31
- operator<
 - ceras, 34
- operator<<
 - ceras, 34
- operator<=
 - ceras, 34
- operator>
 - ceras, 34
- operator>=
 - ceras, 35
- operator*
 - ceras, 31, 32
- operator*=
 - ceras::tensor< T, Allocator >, 88
- operator()
 - ceras::model< Ex, Ph >, 68
- operator+
 - ceras, 32, 33
- operator+=
 - ceras::tensor< T, Allocator >, 88, 89
- operator-
 - ceras, 33
 - ceras::tensor< T, Allocator >, 89
- operator-=
 - ceras::tensor< T, Allocator >, 89
- operator/

- ceras, [33](#)
- operator/=
 - ceras::tensor< T, Allocator >, [89](#)
- operator=
 - ceras::place_holder< T, Allocator >, [72](#)
 - ceras::session< T, Allocator >, [78](#)
 - ceras::tensor< T, Allocator >, [90](#)
 - ceras::value< T >, [98](#)
 - ceras::variable< T, Allocator >, [101](#)
- operator==
 - ceras, [34](#)
- operator[]
 - ceras::tensor< T, Allocator >, [90](#)
 - ceras::view_2d< T >, [104](#), [105](#)
 - ceras::view_3d< T >, [107](#)
 - ceras::view_4d< T >, [110](#)
- output_data_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [58](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [95](#)
- Place_Holder
 - ceras, [46](#)
- place_holder
 - ceras::place_holder< T, Allocator >, [71](#)
- place_holder_
 - ceras::model< Ex, Ph >, [70](#)
- place_holder_type
 - ceras::session< T, Allocator >, [76](#)
- place_holders_
 - ceras::session< T, Allocator >, [79](#)
- plus
 - ceras, [35](#)
- predict
 - ceras::model< Ex, Ph >, [69](#)
- randn
 - ceras, [35](#)
- randn_like
 - ceras, [35](#)
- random
 - ceras, [35](#)
- random_generator
 - ceras, [46](#)
- random_like
 - ceras, [36](#)
- random_normal_like
 - ceras, [36](#)
- random_seed
 - ceras, [46](#)
- read_tensor
 - ceras, [36](#)
- rebind
 - ceras::session< T, Allocator >, [78](#)
- reduce
 - ceras, [36](#)
- reduce_mean
 - ceras, [37](#)
- reduce_sum
 - ceras, [37](#)
- relu
 - ceras, [37](#)
- remember
 - ceras::session< T, Allocator >, [78](#)
- repeat
 - ceras, [37](#)
- replace_placeholder_with_expression
 - ceras, [38](#)
- repmat
 - ceras, [38](#)
- reset
 - ceras::place_holder< T, Allocator >, [72](#)
 - ceras::tensor< T, Allocator >, [90](#)
 - ceras::variable< T, Allocator >, [101](#)
- reset_action_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [59](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [95](#)
- reset_states
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [57](#)
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [94](#)
 - ceras::variable< T, Allocator >, [101](#)
- reshape
 - ceras, [38](#)
 - ceras::tensor< T, Allocator >, [90](#)
- resize
 - ceras::tensor< T, Allocator >, [91](#)
- restore
 - ceras::session< T, Allocator >, [78](#)
- rho_
 - ceras::adadelta< Loss, T >, [51](#)
 - ceras::rmsprop< Loss, T >, [75](#)
- rhs_input_data_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [59](#)
- rhs_op_
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, [59](#)
- rms_prop
 - ceras, [17](#)
- rmsprop
 - ceras::rmsprop< Loss, T >, [74](#)
- row_
 - ceras::view_2d< T >, [105](#)
 - ceras::view_3d< T >, [107](#)
 - ceras::view_4d< T >, [111](#)
- run
 - ceras::session< T, Allocator >, [79](#)
- save
 - ceras::session< T, Allocator >, [79](#)
- save_tensor
 - ceras, [39](#)
- self_type

- ceras::tensor< T, Allocator >, 84
- selu
 - ceras, 39
- serialize
 - ceras::session< T, Allocator >, 79
- session
 - ceras::session< T, Allocator >, 77
- sgd
 - ceras::sgd< Loss, T >, 81
- shape
 - ceras::constant< T, Allocator >, 60
 - ceras::tensor< T, Allocator >, 91
 - ceras::variable< T, Allocator >, 101
 - ceras::view_2d< T >, 105
- shape_
 - ceras::tensor< T, Allocator >, 92
- shape_hint_
 - ceras::place_holder_state< T, Allocator >, 73
- shared_vector
 - ceras::tensor< T, Allocator >, 84
- shrink_to
 - ceras::tensor< T, Allocator >, 91
- sigmoid
 - ceras, 39
- size
 - ceras::tensor< T, Allocator >, 91
 - ceras::view_2d< T >, 105
- slice
 - ceras::tensor< T, Allocator >, 91
- softmax
 - ceras, 39
- softplus
 - ceras, 39
- softsign
 - ceras, 40
- square
 - ceras, 40
- squared_loss
 - ceras, 40
- squeeze
 - ceras, 40
- standard_deviation
 - ceras, 40
- state_
 - ceras::variable< T, Allocator >, 102
- stateful_
 - ceras::variable< T, Allocator >, 102
- STB_IMAGE_IMPLEMENTATION
 - includes.hpp, 116
- STB_IMAGE_RESIZE_IMPLEMENTATION
 - includes.hpp, 116
- STB_IMAGE_WRITE_IMPLEMENTATION
 - includes.hpp, 116
- sum
 - ceras, 40, 41
- sum_reduce
 - ceras, 41
- tanh
 - ceras, 41
- tap
 - ceras::session< T, Allocator >, 79
- Tensor
 - ceras, 47
- tensor
 - ceras::tensor< T, Allocator >, 84, 85
- tensor_type
 - ceras::adadelta< Loss, T >, 49
 - ceras::adagrad< Loss, T >, 51
 - ceras::adam< Loss, T >, 54
 - ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 56
 - ceras::gradient_descent< Loss, T >, 61
 - ceras::place_holder< T, Allocator >, 71
 - ceras::rmsprop< Loss, T >, 74
 - ceras::sgd< Loss, T >, 80
 - ceras::tensor_deduction< L, R >, 92
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 95
 - ceras::variable< T, Allocator >, 99
- tesseract
 - ceras, 17
- trainable_
 - ceras::variable< T, Allocator >, 102
- transpose
 - ceras, 41
- transposed_
 - ceras::view_2d< T >, 106
- truncated_normal
 - ceras, 41
- Unary_Operator
 - ceras, 47
- unary_operator
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 94
- up_sampling_2d
 - ceras, 41
- update_cuda_gemm_threshold
 - ceras, 42
- Value
 - ceras, 47
- value
 - ceras::value< T >, 97
- value_type
 - ceras::tensor< T, Allocator >, 84
 - ceras::value< T >, 96
- Variable
 - ceras, 47
- variable
 - ceras::variable< T, Allocator >, 99, 100
- variable_state_type
 - ceras::session< T, Allocator >, 76
- variable_type
 - ceras::session< T, Allocator >, 76
- variables_
 - ceras::session< T, Allocator >, 79

- variance
 - ceras, [42](#)
- vector_
 - ceras::tensor< T, Allocator >, [92](#)
- vector_type
 - ceras::tensor< T, Allocator >, [84](#)
- version
 - ceras, [47](#)
- view_2d
 - ceras::view_2d< T >, [103](#), [104](#)
- view_3d
 - ceras::view_3d< T >, [106](#)
- view_4d
 - ceras::view_4d< T >, [108](#)
- visible_device
 - ceras, [47](#)
- wrapped_lhs_operator_type
 - ceras::binary_operator< Lhs_Operator, Rh_Operator, Forward_Action, Backward_Action >, [56](#)
- wrapped_operator_type
 - ceras::unary_operator< Operator, Forward_Action, Backward_Action >, [93](#)
- wrapped_rhs_operator_type
 - ceras::binary_operator< Lhs_Operator, Rh_Operator, Forward_Action, Backward_Action >, [56](#)
- write_tensor
 - ceras, [42](#)
- zeros
 - ceras, [42](#)
- zeros_like
 - ceras, [42](#)