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 default_allocator . . . 16

5.1.1.4 rms_prop . . . 16

5.1.2 Function Documentation . . . 16

5.1.2.1 abs() [1/2] . . . 16

5.1.2.2 abs() [2/2] . . . 17

5.1.2.3 abs_loss() . . . 17

5.1.2.4 add() . . . 17

5.1.2.5 amax() . . . 17

5.1.2.6 amin() . . . 17

5.1.2.7 as_tensor() . . . 17

5.1.2.8 average_pooling_2d() . . . 18

5.1.2.9 batch_normalization() . . . 18

5.1.2.10 BatchNormalization() . . . 18

5.1.2.11 clip() [1/2] . . . 18

5.1.2.12 clip() [2/2] . . . 18

5.1.2.13 concat() [1/2] . . . 18

5.1.2.14 concat() [2/2] . . . 19

5.1.2.15 concatenate() [1/3] . . . 19

5.1.2.16 concatenate() [2/3] . . . 19

5.1.2.17 concatenate() [3/3] . . . 19

5.1.2.18 Conv2D() . . . 19

5.1.2.19 conv2d() . . . 20

5.1.2.20 copy() [1/2] . . . 20

5.1.2.21 copy() [2/2] . . . 20

5.1.2.22 cross_entropy() . . . 20

5.1.2.23 cross_entropy_loss() . . . 20

5.1.2.24 deep_copy() . . . 21

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

5.1.2.67 min() [1/2] . . . 28
5.1.2.68 min() [2/2] . . . 28
5.1.2.69 minus() [1/2] . . . 28
5.1.2.70 minus() [2/2] . . . 29
5.1.2.71 mse() . . . 29
5.1.2.72 multiply() [1/2] . . . 29
5.1.2.73 multiply() [2/2] . . . 29
5.1.2.74 negative() . . . 29
5.1.2.75 negative_relu() . . . 29
5.1.2.76 norm() . . . 30
5.1.2.77 normalization_batch() . . . 30
5.1.2.78 normalization_instance() . . . 30
5.1.2.79 ones() . . . 30
5.1.2.80 ones_like() . . . 30
5.1.2.81 operator∗() [1/4] . . . 30
5.1.2.82 operator∗() [2/4] . . . 31
5.1.2.83 operator∗() [3/4] . . . 31
5.1.2.84 operator∗() [4/4] . . . 31
5.1.2.85 operator+() [1/4] . . . 31
5.1.2.86 operator+() [2/4] . . . 31
5.1.2.87 operator+() [3/4] . . . 31
5.1.2.88 operator+() [4/4] . . . 32
5.1.2.89 operator-() [1/4] . . . 32
5.1.2.90 operator-() [2/4] . . . 32
5.1.2.91 operator-() [3/4] . . . 32
5.1.2.92 operator-() [4/4] . . . 32
5.1.2.93 operator/() . . . 32
5.1.2.94 operator<<() . . . 33
5.1.2.95 operator==() . . . 33
5.1.2.96 plus() . . . 33
5.1.2.97 randn() . . . 33
5.1.2.98 randn_like() . . . 33
5.1.2.99 random() . . . 34
5.1.2.100 random_like() . . . 34
5.1.2.101 random_normal_like() . . . 34
5.1.2.102 read_tensor() . . . 34
5.1.2.103 reduce() . . . 35
5.1.2.104 reduce_mean() [1/2] . . . 35
5.1.2.105 reduce_mean() [2/2] . . . 35
5.1.2.106 reduce_sum() [1/2] . . . 35
5.1.2.107 reduce_sum() [2/2] . . . 35
5.1.2.108 relu() . . . 35

5.1.2.109 repeat() . . . 36
5.1.2.110 repmat() . . . 36
5.1.2.111 reshape() [1/2] . . . 36
5.1.2.112 reshape() [2/2] . . . 36
5.1.2.113 save_tensor() . . . 36
5.1.2.114 selu() . . . 36
5.1.2.115 sigmoid() . . . 37
5.1.2.116 softmax() [1/2] . . . 37
5.1.2.117 softmax() [2/2] . . . 37
5.1.2.118 softplus() . . . 37
5.1.2.119 softsign() . . . 37
5.1.2.120 square() . . . 37
5.1.2.121 squared_loss() . . . 38
5.1.2.122 squeeze() . . . 38
5.1.2.123 standard_deviation() . . . 38
5.1.2.124 sum() [1/2] . . . 38
5.1.2.125 sum() [2/2] . . . 38
5.1.2.126 sum_reduce() . . . 38
5.1.2.127 tanh() . . . 39
5.1.2.128 transpose() . . . 39
5.1.2.129 truncated_normal() . . . 39
5.1.2.130 up_sampling_2d() . . . 39
5.1.2.131 update_cuda_gemm_threshold() . . . 39
5.1.2.132 variance() . . . 39
5.1.2.133 write_tensor() . . . 40
5.1.2.134 zeros() . . . 40
5.1.2.135 zeros_like() . . . 40
5.1.3 Variable Documentation . . . 40
5.1.3.1 __version__ . . . 40
5.1.3.2 blas_mode . . . 40
5.1.3.3 Constant . . . 40
5.1.3.4 cuda_gemm_threshold . . . 41
5.1.3.5 cuda_mode . . . 41
5.1.3.6 debug_mode . . . 41
5.1.3.7 eps . . . 41
5.1.3.8 Expression . . . 41
5.1.3.9 is_constant_v . . . 41
5.1.3.10 is_operator_v . . . 41
5.1.3.11 is_place_holder_v . . . 42
5.1.3.12 is_tensor_v . . . 42
5.1.3.13 is_value_v . . . 42
5.1.3.14 is_variable_v . . . 42

5.1.3.15 is_windows_platform . . . 42
5.1.3.16 learning_phase . . . 42
5.1.3.17 lstm . . . 43
5.1.3.18 make_binary_operator . . . 43
5.1.3.19 make_unary_operator . . . 43
5.1.3.20 Operator . . . 43
5.1.3.21 Place_Holder . . . 43
5.1.3.22 random_generator . . . 44
5.1.3.23 random_seed . . . 44
5.1.3.24 Tensor . . . 44
5.1.3.25 Value . . . 44
5.1.3.26 Variable . . . 44
5.1.3.27 version . . . 44
5.1.3.28 visible_device . . . 44

6 Class Documentation **45**

6.1 ceras::adadelta< Loss, T > Struct Template Reference . . . 45
6.1.1 Member Typedef Documentation . . . 45
6.1.1.1 tensor_type . . . 46
6.1.2 Constructor & Destructor Documentation . . . 46
6.1.2.1 adadelta() . . . 46
6.1.3 Member Function Documentation . . . 46
6.1.3.1 forward() . . . 46
6.1.4 Member Data Documentation . . . 46
6.1.4.1 iterations_ . . . 46
6.1.4.2 learning_rate_ . . . 46
6.1.4.3 loss_ . . . 47
6.1.4.4 rho_ . . . 47
6.2 ceras::adagrad< Loss, T > Struct Template Reference . . . 47
6.2.1 Member Typedef Documentation . . . 47
6.2.1.1 tensor_type . . . 48
6.2.2 Constructor & Destructor Documentation . . . 48
6.2.2.1 adagrad() . . . 48
6.2.3 Member Function Documentation . . . 48
6.2.3.1 forward() . . . 48
6.2.4 Member Data Documentation . . . 48
6.2.4.1 decay_ . . . 48
6.2.4.2 iterations_ . . . 48
6.2.4.3 learning_rate_ . . . 49
6.2.4.4 loss_ . . . 49
6.3 ceras::adam< Loss, T > Struct Template Reference . . . 49
6.3.1 Member Typedef Documentation . . . 50

6.3.1.1 tensor_type . . . 50
6.3.2 Constructor & Destructor Documentation . . . 50
6.3.2.1 adam() . . . 50
6.3.3 Member Function Documentation . . . 50
6.3.3.1 forward() . . . 50
6.3.4 Member Data Documentation . . . 50
6.3.4.1 amsgrad_ . . . 50
6.3.4.2 beta_1_ . . . 51
6.3.4.3 beta_2_ . . . 51
6.3.4.4 iterations_ . . . 51
6.3.4.5 learning_rate_ . . . 51
6.3.4.6 loss_ . . . 51
6.4 ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > Struct Template Reference . . . 51
6.4.1 Member Typedef Documentation . . . 52
6.4.1.1 tensor_type . . . 52
6.4.2 Constructor & Destructor Documentation . . . 52
6.4.2.1 binary_operator() . . . 52
6.4.3 Member Function Documentation . . . 53
6.4.3.1 backward() . . . 53
6.4.3.2 forward() . . . 53
6.4.3.3 reset_states() . . . 53
6.4.4 Member Data Documentation . . . 53
6.4.4.1 backward_action_ . . . 53
6.4.4.2 forward_action_ . . . 53
6.4.4.3 lhs_input_data_ . . . 54
6.4.4.4 lhs_op_ . . . 54
6.4.4.5 output_data_ . . . 54
6.4.4.6 reset_action_ . . . 54
6.4.4.7 rhs_input_data_ . . . 54
6.4.4.8 rhs_op_ . . . 55
6.5 ceras::constant< Tsor > Struct Template Reference . . . 55
6.5.1 Constructor & Destructor Documentation . . . 55
6.5.1.1 constant() . . . 55
6.5.2 Member Function Documentation . . . 55
6.5.2.1 backward() . . . 55
6.5.2.2 forward() . . . 56
6.5.2.3 shape() . . . 56
6.5.3 Member Data Documentation . . . 56
6.5.3.1 data_ . . . 56
6.6 ceras::gradient_descent< Loss, T > Struct Template Reference . . . 56
6.6.1 Member Typedef Documentation . . . 57

6.6.1.1 tensor_type . . . 57
6.6.2 Constructor & Destructor Documentation . . . 57
6.6.2.1 gradient_descent() . . . 57
6.6.3 Member Function Documentation . . . 57
6.6.3.1 forward() . . . 57
6.6.4 Member Data Documentation . . . 57
6.6.4.1 learning_rate_ . . . 58
6.6.4.2 loss_ . . . 58
6.6.4.3 momentum_ . . . 58
6.7 ceras::is_constant< T > Struct Template Reference . . . 58
6.8 ceras::is_constant< constant< Tsor > > Struct Template Reference . . . 59
6.9 ceras::is_operator< T > Struct Template Reference . . . 59
6.10 ceras::is_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > > Struct Template Reference . . . 59
6.11 ceras::is_operator< unary_operator< Operator, Forward_Action, Backward_Action > > Struct Template Reference . . . 60
6.12 ceras::is_place_holder< T > Struct Template Reference . . . 60
6.13 ceras::is_place_holder< place_holder< Tsor > > Struct Template Reference . . . 60
6.14 ceras::is_tensor< T > Struct Template Reference . . . 61
6.15 ceras::is_tensor< tensor< T, A > > Struct Template Reference . . . 61
6.16 ceras::is_value< T > Struct Template Reference . . . 61
6.17 ceras::is_value< value< T > > Struct Template Reference . . . 62
6.18 ceras::is_variable< T > Struct Template Reference . . . 62
6.19 ceras::is_variable< variable< Tsor > > Struct Template Reference . . . 62
6.20 ceras::model< Ex, Ph > Struct Template Reference . . . 63
6.20.1 Detailed Description . . . 63
6.20.2 Constructor & Destructor Documentation . . . 63
6.20.2.1 model() . . . 63
6.20.3 Member Function Documentation . . . 64
6.20.3.1 operator()() . . . 64
6.20.3.2 predict() . . . 64
6.20.4 Member Data Documentation . . . 65
6.20.4.1 expression_ . . . 65
6.20.4.2 place_holders_ . . . 65
6.21 ceras::place_holder< Tsor > Struct Template Reference . . . 65
6.21.1 Member Typedef Documentation . . . 66
6.21.1.1 tensor_type . . . 66
6.21.2 Constructor & Destructor Documentation . . . 66
6.21.2.1 place_holder() [1/4] . . . 66
6.21.2.2 place_holder() [2/4] . . . 66
6.21.2.3 place_holder() [3/4] . . . 66
6.21.2.4 place_holder() [4/4] . . . 67
6.21.3 Member Function Documentation . . . 67

6.21.3.1 backward() . . . 67
6.21.3.2 bind() . . . 67
6.21.3.3 forward() . . . 67
6.21.3.4 operator=() [1/2] . . . 67
6.21.3.5 operator=() [2/2] . . . 67
6.21.3.6 reset() . . . 68
6.22 ceras::place_holder_state< Tsor > Struct Template Reference . . . 68
6.22.1 Member Data Documentation . . . 68
6.22.1.1 data_ . . . 68
6.22.1.2 shape_hint_ . . . 68
6.23 ceras::rmsprop< Loss, T > Struct Template Reference . . . 68
6.23.1 Member Typedef Documentation . . . 69
6.23.1.1 tensor_type . . . 69
6.23.2 Constructor & Destructor Documentation . . . 69
6.23.2.1 rmsprop() . . . 69
6.23.3 Member Function Documentation . . . 69
6.23.3.1 forward() . . . 70
6.23.4 Member Data Documentation . . . 70
6.23.4.1 decay_ . . . 70
6.23.4.2 iterations_ . . . 70
6.23.4.3 learning_rate_ . . . 70
6.23.4.4 loss_ . . . 70
6.23.4.5 rho_ . . . 70
6.24 ceras::session< Tsor > Struct Template Reference . . . 71
6.24.1 Member Typedef Documentation . . . 71
6.24.1.1 place_holder_type . . . 71
6.24.1.2 variable_state_type . . . 72
6.24.1.3 variable_type . . . 72
6.24.2 Constructor & Destructor Documentation . . . 72
6.24.2.1 session() [1/3] . . . 72
6.24.2.2 session() [2/3] . . . 72
6.24.2.3 session() [3/3] . . . 72
6.24.2.4 ~session() . . . 72
6.24.3 Member Function Documentation . . . 73
6.24.3.1 bind() . . . 73
6.24.3.2 deserialize() . . . 73
6.24.3.3 operator=() [1/2] . . . 73
6.24.3.4 operator=() [2/2] . . . 73
6.24.3.5 rebind() . . . 73
6.24.3.6 remember() . . . 74
6.24.3.7 restore() . . . 74
6.24.3.8 run() . . . 74

6.24.3.9 save() . . . 74
6.24.3.10 serialize() . . . 74
6.24.3.11 tap() . . . 74
6.24.4 Member Data Documentation . . . 75
6.24.4.1 place_holders_ . . . 75
6.24.4.2 variables_ . . . 75
6.25 ceras::sgd< Loss, T > Struct Template Reference . . . 75
6.25.1 Member Typedef Documentation . . . 76
6.25.1.1 tensor_type . . . 76
6.25.2 Constructor & Destructor Documentation . . . 76
6.25.2.1 sgd() . . . 76
6.25.3 Member Function Documentation . . . 76
6.25.3.1 forward() . . . 76
6.25.4 Member Data Documentation . . . 76
6.25.4.1 decay_ . . . 77
6.25.4.2 iterations_ . . . 77
6.25.4.3 learning_rate_ . . . 77
6.25.4.4 loss_ . . . 77
6.25.4.5 momentum_ . . . 77
6.25.4.6 nesterov_ . . . 77
6.26 ceras::tensor< T, Allocator > Struct Template Reference . . . 78
6.26.1 Member Typedef Documentation . . . 79
6.26.1.1 allocator . . . 79
6.26.1.2 self_type . . . 79
6.26.1.3 shared_vector . . . 79
6.26.1.4 value_type . . . 80
6.26.1.5 vector_type . . . 80
6.26.2 Constructor & Destructor Documentation . . . 80
6.26.2.1 tensor() [1/7] . . . 80
6.26.2.2 tensor() [2/7] . . . 80
6.26.2.3 tensor() [3/7] . . . 80
6.26.2.4 tensor() [4/7] . . . 80
6.26.2.5 tensor() [5/7] . . . 81
6.26.2.6 tensor() [6/7] . . . 81
6.26.2.7 tensor() [7/7] . . . 81
6.26.3 Member Function Documentation . . . 81
6.26.3.1 begin() [1/2] . . . 81
6.26.3.2 begin() [2/2] . . . 81
6.26.3.3 cbegin() . . . 81
6.26.3.4 cend() . . . 82
6.26.3.5 copy() . . . 82
6.26.3.6 creep_to() . . . 82

6.26.3.7 data() [1/2] . . . 82
6.26.3.8 data() [2/2] . . . 82
6.26.3.9 deep_copy() [1/2] . . . 82
6.26.3.10 deep_copy() [2/2] . . . 83
6.26.3.11 empty() . . . 83
6.26.3.12 end() [1/2] . . . 83
6.26.3.13 end() [2/2] . . . 83
6.26.3.14 map() . . . 83
6.26.3.15 ndim() . . . 83
6.26.3.16 operator*=() [1/2] . . . 84
6.26.3.17 operator*=() [2/2] . . . 84
6.26.3.18 operator+=() [1/2] . . . 84
6.26.3.19 operator+=() [2/2] . . . 84
6.26.3.20 operator-() . . . 84
6.26.3.21 operator-=() [1/2] . . . 84
6.26.3.22 operator-=() [2/2] . . . 85
6.26.3.23 operator/=() [1/2] . . . 85
6.26.3.24 operator/=() [2/2] . . . 85
6.26.3.25 operator=() [1/2] . . . 85
6.26.3.26 operator=() [2/2] . . . 85
6.26.3.27 operator[]() [1/2] . . . 85
6.26.3.28 operator[]() [2/2] . . . 86
6.26.3.29 reset() . . . 86
6.26.3.30 reshape() . . . 86
6.26.3.31 resize() . . . 86
6.26.3.32 shape() . . . 86
6.26.3.33 shrink_to() . . . 86
6.26.3.34 size() . . . 87
6.26.3.35 slice() . . . 87
6.26.4 Member Data Documentation . . . 87
6.26.4.1 memory_offset_ . . . 87
6.26.4.2 shape_ . . . 87
6.26.4.3 vector_ . . . 87
6.27 ceras::tensor_deduction< L, R > Struct Template Reference . . . 87
6.27.1 Member Typedef Documentation . . . 88
6.27.1.1 op_type . . . 88
6.27.1.2 tensor_type . . . 88
6.28 ceras::unary_operator< Operator, Forward_Action, Backward_Action > Struct Template Reference 88
6.28.1 Constructor & Destructor Documentation . . . 89
6.28.1.1 unary_operator() . . . 89
6.28.2 Member Function Documentation . . . 89
6.28.2.1 backward() . . . 89

6.28.2.2 forward() . . . 89
6.28.2.3 reset_states() . . . 89
6.28.3 Member Data Documentation . . . 90
6.28.3.1 backward_action_ . . . 90
6.28.3.2 forward_action_ . . . 90
6.28.3.3 input_data_ . . . 90
6.28.3.4 op_ . . . 90
6.28.3.5 output_data_ . . . 90
6.28.3.6 reset_action_ . . . 90
6.28.3.7 tensor_type . . . 91
6.29 ceras::value< T > Struct Template Reference . . . 91
6.29.1 Member Typedef Documentation . . . 91
6.29.1.1 value_type . . . 91
6.29.2 Constructor & Destructor Documentation . . . 92
6.29.2.1 value() [1/4] . . . 92
6.29.2.2 value() [2/4] . . . 92
6.29.2.3 value() [3/4] . . . 92
6.29.2.4 value() [4/4] . . . 92
6.29.3 Member Function Documentation . . . 92
6.29.3.1 backward() . . . 92
6.29.3.2 forward() . . . 93
6.29.3.3 operator=() [1/2] . . . 93
6.29.3.4 operator=() [2/2] . . . 93
6.29.4 Member Data Documentation . . . 93
6.29.4.1 data_ . . . 93
6.30 ceras::variable< Tsor > Struct Template Reference . . . 93
6.30.1 Member Typedef Documentation . . . 94
6.30.1.1 tensor_type . . . 94
6.30.2 Constructor & Destructor Documentation . . . 94
6.30.2.1 variable() [1/4] . . . 94
6.30.2.2 variable() [2/4] . . . 95
6.30.2.3 variable() [3/4] . . . 95
6.30.2.4 variable() [4/4] . . . 95
6.30.3 Member Function Documentation . . . 95
6.30.3.1 backward() . . . 95
6.30.3.2 data() [1/2] . . . 95
6.30.3.3 data() [2/2] . . . 95
6.30.3.4 forward() . . . 96
6.30.3.5 gradient() [1/2] . . . 96
6.30.3.6 gradient() [2/2] . . . 96
6.30.3.7 operator=() [1/2] . . . 96
6.30.3.8 operator=() [2/2] . . . 96

6.30.3.9 reset() . . . 96
6.30.3.10 reset_states() . . . 96
6.30.3.11 shape() . . . 97
6.30.4 Member Data Documentation . . . 97
6.30.4.1 state_ . . . 97
6.30.4.2 stateful_ . . . 97
6.30.4.3 trainable_ . . . 97
6.31 ceras::variable_state< Tsor > Struct Template Reference . . . 97
6.31.1 Member Data Documentation . . . 97
6.31.1.1 data_ . . . 98
6.31.1.2 gradient_ . . . 98
6.32 ceras::view_2d< T > Struct Template Reference . . . 98
6.32.1 Constructor & Destructor Documentation . . . 98
6.32.1.1 view_2d() [1/3] . . . 99
6.32.1.2 view_2d() [2/3] . . . 99
6.32.1.3 view_2d() [3/3] . . . 99
6.32.2 Member Function Documentation . . . 99
6.32.2.1 data() [1/2] . . . 99
6.32.2.2 data() [2/2] . . . 99
6.32.2.3 operator[]() [1/2] . . . 100
6.32.2.4 operator[]() [2/2] . . . 100
6.32.2.5 shape() . . . 100
6.32.2.6 size() . . . 100
6.32.3 Member Data Documentation . . . 100
6.32.3.1 col_ . . . 100
6.32.3.2 data_ . . . 100
6.32.3.3 row_ . . . 101
6.32.3.4 transposed_ . . . 101
6.33 ceras::view_3d< T > Struct Template Reference . . . 101
6.33.1 Constructor & Destructor Documentation . . . 101
6.33.1.1 view_3d() . . . 101
6.33.2 Member Function Documentation . . . 102
6.33.2.1 operator[]() [1/2] . . . 102
6.33.2.2 operator[]() [2/2] . . . 102
6.33.3 Member Data Documentation . . . 102
6.33.3.1 channel_ . . . 102
6.33.3.2 col_ . . . 102
6.33.3.3 data_ . . . 102
6.33.3.4 row_ . . . 103
6.34 ceras::view_4d< T > Struct Template Reference . . . 103
6.34.1 Detailed Description . . . 103
6.34.2 Constructor & Destructor Documentation . . . 103

6.34.2.1 view_4d() . . . 104
6.34.3 Member Function Documentation . . . 105
6.34.3.1 operator[]() [1/2] . . . 105
6.34.3.2 operator[]() [2/2] . . . 105
6.34.4 Member Data Documentation . . . 106
6.34.4.1 batch_size_ . . . 106
6.34.4.2 channel_ . . . 106
6.34.4.3 col_ . . . 106
6.34.4.4 data_ . . . 106
6.34.4.5 row_ . . . 106

7 File Documentation **107**

7.1 /data/structured_folders/workspace/github.repo/ceras/include/activation.hpp File Reference . . . 107
7.2 /data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp File Reference . . . 108
7.3 /data/structured_folders/workspace/github.repo/ceras/include/config.hpp File Reference . . . 108
7.4 /data/structured_folders/workspace/github.repo/ceras/include/constant.hpp File Reference . . . 109
7.5 /data/structured_folders/workspace/github.repo/ceras/include/includes.hpp File Reference . . . 109
7.5.1 Macro Definition Documentation . . . 110
7.5.1.1 STB_IMAGE_IMPLEMENTATION . . . 110
7.5.1.2 STB_IMAGE_RESIZE_IMPLEMENTATION . . . 110
7.5.1.3 STB_IMAGE_WRITE_IMPLEMENTATION . . . 110
7.6 /data/structured_folders/workspace/github.repo/ceras/include/keras.hpp File Reference . . . 111
7.7 /data/structured_folders/workspace/github.repo/ceras/include/layer.hpp File Reference . . . 111
7.8 /data/structured_folders/workspace/github.repo/ceras/include/loss.hpp File Reference . . . 111
7.9 /data/structured_folders/workspace/github.repo/ceras/include/model.hpp File Reference . . . 112
7.10 /data/structured_folders/workspace/github.repo/ceras/include/operation.hpp File Reference . . . 113
7.11 /data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp File Reference . . . 115
7.12 /data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp File Reference . . . 116
7.13 /data/structured_folders/workspace/github.repo/ceras/include/recurrent_operation.hpp File Reference 116
7.14 /data/structured_folders/workspace/github.repo/ceras/include/session.hpp File Reference . . . 117
7.15 /data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp File Reference . . . 117
7.16 /data/structured_folders/workspace/github.repo/ceras/include/value.hpp File Reference . . . 121
7.17 /data/structured_folders/workspace/github.repo/ceras/include/variable.hpp File Reference . . . 122

Index **123**

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< Tsor > . . . 55
enable_id
  ceras::adadelta< Loss, T > . . . 45
  ceras::adagrad< Loss, T > . . . 47
  ceras::adam< Loss, T > . . . 49
  ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > . . . 51
  ceras::gradient_descent< Loss, T > . . . 56
  ceras::place_holder< Tsor > . . . 65
  ceras::rmsprop< Loss, T > . . . 68
  ceras::sgd< Loss, T > . . . 75
  ceras::tensor< T, Allocator > . . . 78
  ceras::unary_operator< Operator, Forward_Action, Backward_Action > . . . 88
  ceras::variable< Tsor > . . . 93
enable_shared
  ceras::adadelta< Loss, T > . . . 45
  ceras::adagrad< Loss, T > . . . 47
  ceras::adam< Loss, T > . . . 49
  ceras::gradient_descent< Loss, T > . . . 56
  ceras::rmsprop< Loss, T > . . . 68
  ceras::sgd< Loss, T > . . . 75
enable_shared_state
  ceras::place_holder< Tsor > . . . 65
std::false_type
  ceras::is_constant< T > . . . 58
  ceras::is_operator< T > . . . 59
  ceras::is_place_holder< T > . . . 60
  ceras::is_tensor< T > . . . 61
  ceras::is_value< T > . . . 61
  ceras::is_variable< T > . . . 62
ceras::model< Ex, Ph > . . . 63
ceras::place_holder_state< Tsor > . . . 68
ceras::session< Tsor > . . . 71
ceras::tensor_deduction< L, R > . . . 87
ceras::tensor_deduction< Lhs_Operator, Rhs_Operator > . . . 87
std::true_type

ceras::is_constant< constant< Tsor > > . . . 59
ceras::is_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_↩
Action > > . . . 59
ceras::is_operator< unary_operator< Operator, Forward_Action, Backward_Action > > . . . 60
ceras::is_place_holder< place_holder< Tsor > > . . . 60
ceras::is_tensor< tensor< T, A > > . . . 61
ceras::is_value< value< T > > . . . 62
ceras::is_variable< variable< Tsor > > . . . 62
ceras::value< T > . . . 91
ceras::variable_state< Tsor > . . . 97
ceras::view_2d< T > . . . 98
ceras::view_3d< T > . . . 101
ceras::view_4d< T > . . . 103

Chapter 3

Class Index

3.1 Class List

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

ceras::adadelta< Loss, T > . . . 45
ceras::adagrad< Loss, T > . . . 47
ceras::adam< Loss, T > . . . 49
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > . . . 51
ceras::constant< Tsor > . . . 55
ceras::gradient_descent< Loss, T > . . . 56
ceras::is_constant< T > . . . 58
ceras::is_constant< constant< Tsor > > . . . 59
ceras::is_operator< T > . . . 59
ceras::is_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > > 59
ceras::is_operator< unary_operator< Operator, Forward_Action, Backward_Action > > . . . 60
ceras::is_place_holder< T > . . . 60
ceras::is_place_holder< place_holder< Tsor > > . . . 60
ceras::is_tensor< T > . . . 61
ceras::is_tensor< tensor< T, A > > . . . 61
ceras::is_value< T > . . . 61
ceras::is_value< value< T > > . . . 62
ceras::is_variable< T > . . . 62
ceras::is_variable< variable< Tsor > > . . . 62
ceras::model< Ex, Ph > . . . 63
ceras::place_holder< Tsor > . . . 65
ceras::place_holder_state< Tsor > . . . 68
ceras::rmsprop< Loss, T > . . . 68
ceras::session< Tsor > . . . 71
ceras::sgd< Loss, T > . . . 75
ceras::tensor< T, Allocator > . . . 78
ceras::tensor_deduction< L, R > . . . 87
ceras::unary_operator< Operator, Forward_Action, Backward_Action > . . . 88
ceras::value< T > . . . 91
ceras::variable< Tsor > . . . 93
ceras::variable_state< Tsor > . . . 97
ceras::view_2d< T > . . . 98
ceras::view_3d< T > . . . 101
ceras::view_4d< T > . . . 103

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 . . . 107
/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp . . . 108
/data/structured_folders/workspace/github.repo/ceras/include/config.hpp . . . 108
/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp . . . 109
/data/structured_folders/workspace/github.repo/ceras/include/includes.hpp . . . 109
/data/structured_folders/workspace/github.repo/ceras/include/keras.hpp . . . 111
/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp . . . 111
/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp . . . 111
/data/structured_folders/workspace/github.repo/ceras/include/model.hpp . . . 112
/data/structured_folders/workspace/github.repo/ceras/include/operation.hpp . . . 113
/data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp . . . 115
/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp . . . 116
/data/structured_folders/workspace/github.repo/ceras/include/recurrent_operation.hpp . . . 116
/data/structured_folders/workspace/github.repo/ceras/include/session.hpp . . . 117
/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp . . . 117
/data/structured_folders/workspace/github.repo/ceras/include/value.hpp . . . 121
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp . . . 122

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_operator
- struct is_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > >
- struct is_operator< unary_operator< Operator, Forward_Action, Backward_Action > >
- struct sgd
- struct adagrad
- struct rmsprop
- struct adadelta
- struct adam
- struct gradient_descent
- struct place_holder_state
- struct place_holder
- struct is_place_holder
- struct is_place_holder< place_holder< Tsor > >
- struct 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 >

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 Rhs_Expression>
  constexpr auto mean_squared_logarithmic_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto squared_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto mean_squared_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto mse (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept

- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto abs_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto mean_absolute_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto mae (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto cross_entropy (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto cross_entropy_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto hinge_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto plus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto operator+ (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  auto operator∗ (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto log (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto negative (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto elementwise_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto hadamard_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto sum_reduce (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto reduce_sum (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto mean_reduce (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto reduce_mean (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto minus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto operator- (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto square (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto abs (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto exp (Ex const &ex) noexcept
- template<typename Float >
  requires std::floating_point< Float > constexpr auto clip (Float lower, Float upper=std::numeric_limits< Float >::max()) noexcept
- auto reshape (std::vector< unsigned long > const &new_shape, bool include_batch_flag=true) noexcept
- template<Expression Ex>
  constexpr auto flatten (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto identity (Ex const &ex) noexcept

- template<Expression Ex>
  auto transpose (Ex const &ex) noexcept
- auto img2col (unsigned long const row_kernel, unsigned long col_kernel=-1, unsigned long const row_↩ 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 conv2d (unsigned long row_input, unsigned long col_input, unsigned long const row_stride=1, unsigned long const col_stride=1, unsigned long const row_dilation=1, unsigned long const col_dilation=1, std::string const &padding="valid") noexcept
- template<typename T >
  requires std::floating_point< T > auto drop_out (T const factor) noexcept
- auto max_pooling_2d (unsigned long stride) noexcept
- auto average_pooling_2d (unsigned long stride) noexcept
- auto up_sampling_2d (unsigned long stride) noexcept
- template<typename T = double>
  requires std::floating_point< T > auto normalization_batch (T const momentum=0.98) noexcept
- template<typename T >
  requires std::floating_point< T > auto batch_normalization (T const momentum=0.98) noexcept
- template<typename T = double>
  requires std::floating_point< T > auto normalization_instance (T const momentum=0.98) noexcept
- template<typename T >
  requires std::floating_point< T > auto instance_normalization (T const momentum=0.98) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto concatenate (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- auto concatenate (unsigned long axe=-1)
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto concat (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- auto concat (unsigned long axe=-1)
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto maximum (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<typename T >
  requires std::floating_point< T > auto random_normal_like (T mean=0.0, T stddev=1.0) noexcept
- template<Expression Lhs_Expression, Variable Rhs_Variable>
  constexpr auto copy (Lhs_Expression const &lhs_ex, Rhs_Variable const &rhs_va) noexcept
- template<Tensor Tsor>
  std::reference_wrapper< session< Tsor > > get_default_session ()
- template<typename T , typename A = default_allocator<T>>
  constexpr tensor< T, A > as_tensor (T val) noexcept
- template<Tensor Tsor, typename CharT , typename Traits >
  std::basic_ostream< CharT, Traits > & operator<< (std::basic_ostream< CharT, Traits > &os_, Tsor const &tsor)
- template<typename T >
  requires std::floating_point< T > void gemm_cpu (T const *A, bool a_transposed, T const *B, bool b_↩ transposed, unsigned long m, unsigned long n, unsigned long k, T *C)
- void update_cuda_gemm_threshold ()
- template<typename T >
  requires std::floating_point< T > void gemm (T const *A, bool a_transposed, T const *B, bool b_transposed, unsigned long m, unsigned long n, unsigned long k, T *C)
- template<typename T >
  requires std::floating_point< T > void gemm (view_2d< T > const &x, view_2d< T > const &y, view_2d< T > &ans)
- template<Tensor Tsor>
  Tsor add (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor operator+ (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor operator+ (typename Tsor::value_type const &lhs, Tsor const &rhs) noexcept

- template<Tensor Tsor>
  Tsor operator+ (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept
- template<Tensor Tsor>
  Tsor minus (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor operator- (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor operator- (typename Tsor::value_type const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor operator- (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept
- template<Tensor Tsor>
  Tsor operator* (typename Tsor::value_type const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor operator* (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept
- template<Tensor Tsor>
  Tsor operator/ (Tsor const &lhs, typename Tsor::value_type const &rhs) noexcept
- template<Tensor Tsor>
  Tsor reshape (Tsor const &ts, std::vector< unsigned long > const &new_shape)
- template<Tensor Tsor>
  void multiply (Tsor const &lhs, Tsor const &rhs, Tsor &ans) noexcept
- template<Tensor Tsor>
  Tsor multiply (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor operator* (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor elementwise_product (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor hadamard_product (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor elementwise_divide (Tsor const &lhs, Tsor const &rhs) noexcept
- template<Tensor Tsor>
  Tsor repeat (Tsor const &tsor, unsigned long n)
- template<Tensor Tsor>
  Tsor reduce_sum (Tsor const &tsor)
- template<Tensor Tsor>
  Tsor reduce_mean (Tsor const &tsor)
- template<Tensor Tsor>
  Tsor clip (Tsor &tsor, typename Tsor::value_type lower=0, typename Tsor::value_type upper=1)
- template<Tensor Tsor>
  Tsor squeeze (Tsor const &tsor)
- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > randn (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1})
- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > truncated_normal (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1}, T lower=T{0}, T upper=T{1})
- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > random (std::vector< unsigned long > const &shape, T min=T{0}, T max=T{1})
- template<Tensor Tsor>
  Tsor random_like (Tsor const &tsor, typename Tsor::value_type min=0, typename Tsor::value_type max=1)
- template<Tensor Tsor>
  Tsor randn_like (Tsor const &tsor, typename Tsor::value_type mean=0, typename Tsor::value_type stddev=1)
- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > 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_operator_v = is_operator<T>::value
- template<typename T >
  concept Operator = is_operator_v<T>
- template<typename T >
  concept Expression = Operator<T> || Variable<T> || Place_Holder<T> || Constant<T> || Value<T>
- template<class T >
  constexpr bool is_place_holder_v = is_place_holder<T>::value
- template<typename T >
  concept Place_Holder = is_place_holder_v<T>
- auto lstm
- static unsigned long random_seed = std::chrono::system_clock::now().time_since_epoch().count()
- static std::mt19937 random_generator {random_seed}
- template<class T >
  constexpr bool is_tensor_v = is_tensor<T>::value
- template<typename T >
  concept Tensor = is_tensor_v<T>
- template<class T >
  constexpr bool is_value_v = is_value<T>::value

- template<typename T >
  concept Value = is_value_v<T>
- template<class T >
  constexpr bool is_variable_v = is_variable<T>::value
- template<typename T >
  concept Variable = is_variable_v<T>

5.1.1 Typedef Documentation

5.1.1.1 ada_delta

```
template<typename Loss , typename T >
using ceras::ada_delta = typedef adadelta< Loss, T >
```

5.1.1.2 ada_grad

```
template<typename Loss , typename T >
using ceras::ada_grad = typedef adagrad<Loss, T>
```

5.1.1.3 default_allocator

```
template<typename T >
using ceras::default_allocator = typedef std::allocator<T>
```

5.1.1.4 rms_prop

```
template<typename Loss , typename T >
using ceras::rms_prop = typedef rmsprop< Loss, 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*() [1/4]

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

5.1.2.82 operator∗() [2/4]

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

5.1.2.83 operator∗() [3/4]

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

5.1.2.84 operator∗() [4/4]

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

5.1.2.85 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]
```

5.1.2.86 operator+() [2/4]

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

5.1.2.87 operator+() [3/4]

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

5.1.2.88 operator+() [4/4]

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

5.1.2.89 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]
```

5.1.2.90 operator-() [2/4]

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

5.1.2.91 operator-() [3/4]

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

5.1.2.92 operator-() [4/4]

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

5.1.2.93 operator/()

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

5.1.2.94 operator<<()

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

5.1.2.95 operator==()

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

5.1.2.96 plus()

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

5.1.2.97 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} )
```

5.1.2.98 randn_like()

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

5.1.2.99 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} )
```

5.1.2.100 random_like()

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

5.1.2.101 random_normal_like()

```
template<typename T >
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

mean	Mean of the normal distribution, a scalar.
stddev	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)
```

5.1.2.102 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 )
```

5.1.2.103 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]
```

5.1.2.104 reduce_mean() [1/2]

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

5.1.2.105 reduce_mean() [2/2]

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

5.1.2.106 reduce_sum() [1/2]

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

5.1.2.107 reduce_sum() [2/2]

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

5.1.2.108 relu()

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

5.1.2.109 repeat()

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

5.1.2.110 repmat()

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

5.1.2.111 reshape() [1/2]

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

5.1.2.112 reshape() [2/2]

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

5.1.2.113 save_tensor()

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

5.1.2.114 selu()

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

5.1.2.115 sigmoid()

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

5.1.2.116 softmax() [1/2]

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

5.1.2.117 softmax() [2/2]

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

5.1.2.118 softplus()

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

5.1.2.119 softsign()

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

5.1.2.120 square()

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

5.1.2.121 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]
```

5.1.2.122 squeeze()

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

5.1.2.123 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]
```

5.1.2.124 sum() [1/2]

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

5.1.2.125 sum() [2/2]

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

5.1.2.126 sum_reduce()

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

5.1.2.127 tanh()

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

5.1.2.128 transpose()

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

5.1.2.129 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} )
```

5.1.2.130 up_sampling_2d()

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

5.1.2.131 update_cuda_gemm_threshold()

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

5.1.2.132 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]
```

5.1.2.133 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 )
```

5.1.2.134 zeros()

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

5.1.2.135 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 blas_mode

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

5.1.3.3 Constant

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

5.1.3.4 cuda_gemm_threshold

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

5.1.3.5 cuda_mode

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

5.1.3.6 debug_mode

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

5.1.3.7 eps

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

5.1.3.8 Expression

```
template<typename T >
concept ceras::Expression = Operator<T> || Variable<T> || Place_Holder<T> || Constant<T> ||
Value<T>
```

5.1.3.9 is_constant_v

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

5.1.3.10 is_operator_v

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

5.1.3.11 is_place_holder_v

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

5.1.3.12 is_tensor_v

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

5.1.3.13 is_value_v

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

5.1.3.14 is_variable_v

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

5.1.3.15 is_windows_platform

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

Parameters

is_windows_platform	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();
}
```

5.1.3.16 learning_phase

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

5.1.3.17 lstm

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

5.1.3.18 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;
        };
    }
```

5.1.3.19 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;
        };
    }
```

5.1.3.20 Operator

```
template<typename T >
concept ceras::Operator = is_operator_v<T>
```

5.1.3.21 Place_Holder

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

5.1.3.22 random_generator

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

5.1.3.23 random_seed

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

5.1.3.24 Tensor

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

5.1.3.25 Value

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

5.1.3.26 Variable

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

5.1.3.27 version

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

5.1.3.28 visible_device

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

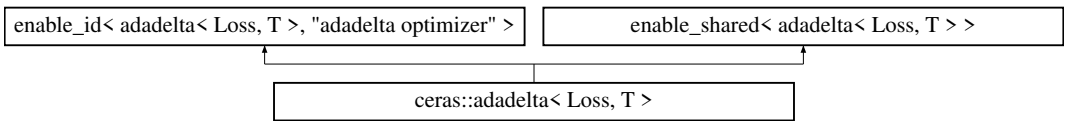
Chapter 6

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::adadelta< Loss, T >::loss_
```

6.1.4.4 rho_

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

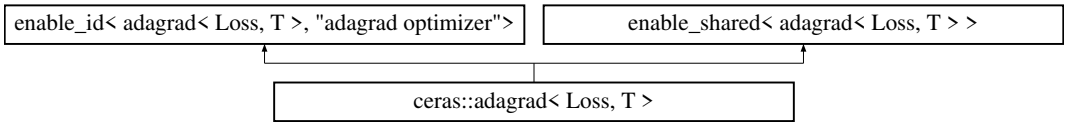
The documentation for this struct was generated from the following file:

- /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_
```

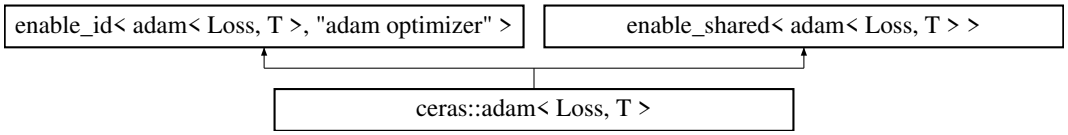
The documentation for this struct was generated from the following file:

- /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_
```

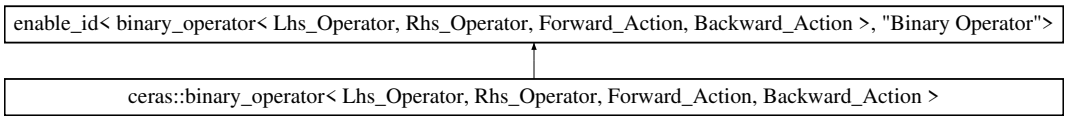
The documentation for this struct was generated from the following file:

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

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

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

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



Public Types

- using tensor_type = typename tensor_deduction< Lhs_Operator, Rhs_Operator >::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 >
using ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >↩
::tensor_type = typename tensor_deduction<Lhs_Operator, Rhs_Operator>::tensor_type
```

6.4.2 Constructor & Destructor Documentation

6.4.2.1 binary_operator()

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >::binary_operator
(
            Lhs_Operator const & lhs_op,
            Rhs_Operator const & rhs_op,
            Forward_Action const & forward_action,
            Backward_Action const & backward_action,
            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

- Tsor 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 Tsor>
Tsor ceras::constant< Tsor >::data_
```

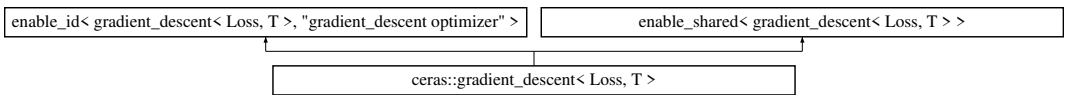
The documentation for this struct was generated from the following file:

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

6.6 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_
```

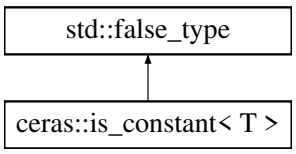
The documentation for this struct was generated from the following file:

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

6.7 ceras::is_constant< T > Struct Template Reference

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

Inheritance diagram for ceras::is_constant< T >:



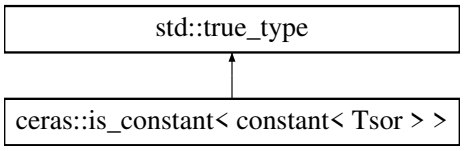
The documentation for this struct was generated from the following file:

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

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

`#include <constant.hpp>`

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



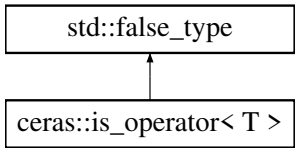
The documentation for this struct was generated from the following file:

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

6.9 ceras::is_operator< T > Struct Template Reference

`#include <operation.hpp>`

Inheritance diagram for ceras::is_operator< T >:



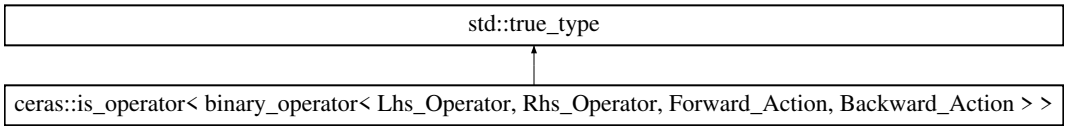
The documentation for this struct was generated from the following file:

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

6.10 ceras::is_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > > Struct Template Reference

`#include <operation.hpp>`

Inheritance diagram for ceras::is_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > >:



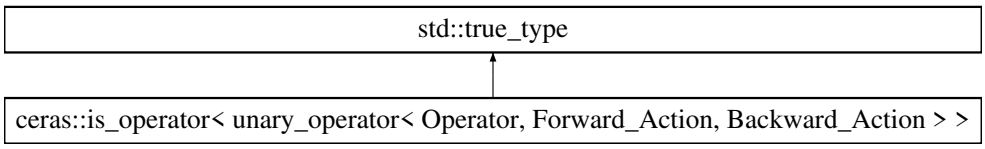
The documentation for this struct was generated from the following file:

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

6.11 ceras::is_operator< unary_operator< Operator, Forward_Action, Backward_Action > > Struct Template Reference

`#include <operation.hpp>`

Inheritance diagram for ceras::is_operator< unary_operator< Operator, Forward_Action, Backward_Action > >:



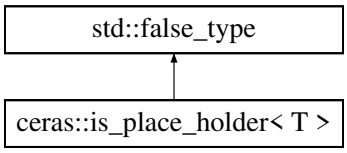
The documentation for this struct was generated from the following file:

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

6.12 ceras::is_place_holder< T > Struct Template Reference

`#include <place_holder.hpp>`

Inheritance diagram for ceras::is_place_holder< T >:



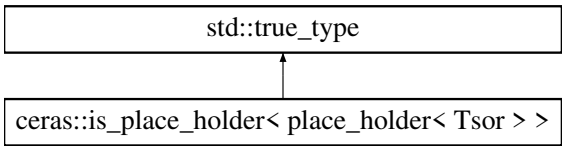
The documentation for this struct was generated from the following file:

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

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

`#include <place_holder.hpp>`

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



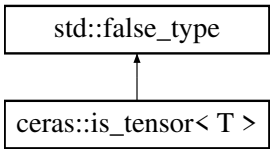
The documentation for this struct was generated from the following file:

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

6.14 ceras::is_tensor< T > Struct Template Reference

`#include <tensor.hpp>`

Inheritance diagram for ceras::is_tensor< T >:



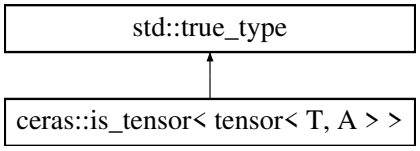
The documentation for this struct was generated from the following file:

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

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

`#include <tensor.hpp>`

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



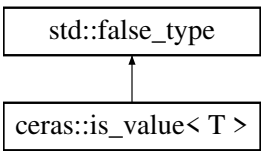
The documentation for this struct was generated from the following file:

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

6.16 ceras::is_value< T > Struct Template Reference

`#include <value.hpp>`

Inheritance diagram for ceras::is_value< T >:



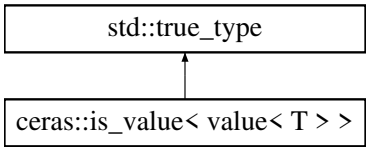
The documentation for this struct was generated from the following file:

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

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

`#include <value.hpp>`

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



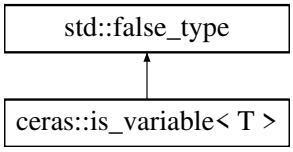
The documentation for this struct was generated from the following file:

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

6.18 ceras::is_variable< T > Struct Template Reference

`#include <variable.hpp>`

Inheritance diagram for ceras::is_variable< T >:



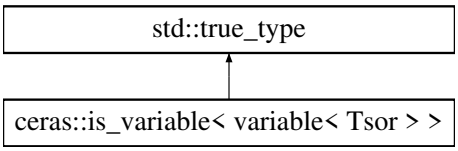
The documentation for this struct was generated from the following file:

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

6.19 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

6.20 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<Tensor Tsor>
  auto operator() (Tsor const &input_tensor)

Public Attributes

- Ex expression_
  output layer of the model.
- std::vector< Ph > place_holders_
  input layer of the model.

6.20.1 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

Ex	The expression template for the output layer.
Ph	The place holder expression for the input layer

6.20.2 Constructor & Destructor Documentation

6.20.2.1 model()

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

Parameters

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

6.20.3 Member Function Documentation

6.20.3.1 operator()()

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

An alias method of `predict`

See also

predict()

Parameters

input_tensor	The input samples.

Returns

The result this model predicts.

6.20.3.2 predict()

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

Parameters

input_tensor	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)
```

6.20.4 Member Data Documentation

6.20.4.1 expression_

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

output layer of the model.

6.20.4.2 place_holders_

```
template<Expression Ex, Place_Holder Ph>
std::vector<Ph> ceras::model< Ex, Ph >::place_holders_
```

input layer of the model.

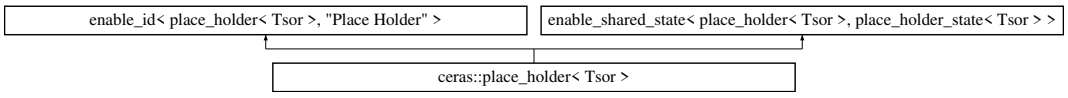
The documentation for this struct was generated from the following file:

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

6.21 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

6.21.1 Member Typedef Documentation

6.21.1.1 tensor_type

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

6.21.2 Constructor & Destructor Documentation

6.21.2.1 place_holder() [1/4]

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

6.21.2.2 place_holder() [2/4]

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

6.21.2.3 place_holder() [3/4]

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

6.21.2.4 place_holder() [4/4]

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

6.21.3 Member Function Documentation

6.21.3.1 backward()

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

6.21.3.2 bind()

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

6.21.3.3 forward()

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

6.21.3.4 operator=() [1/2]

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

6.21.3.5 operator=() [2/2]

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

6.21.3.6 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

6.22 ceras::place_holder_state< Tsor > Struct Template Reference

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

Public Attributes

- Tsor data_
- std::vector< unsigned long > shape_hint_

6.22.1 Member Data Documentation

6.22.1.1 data_

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

6.22.1.2 shape_hint_

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

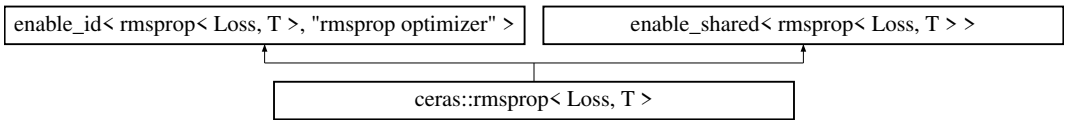
The documentation for this struct was generated from the following file:

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

6.23 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_

6.23.1 Member Typedef Documentation

6.23.1.1 tensor_type

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

6.23.2 Constructor & Destructor Documentation

6.23.2.1 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]
```

6.23.3 Member Function Documentation

6.23.3.1 forward()

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

6.23.4 Member Data Documentation

6.23.4.1 decay_

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

6.23.4.2 iterations_

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

6.23.4.3 learning_rate_

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

6.23.4.4 loss_

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

6.23.4.5 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

6.24 ceras::session< Tsor > Struct Template Reference

`#include <session.hpp>`

Public Types

- typedef place_holder< Tsor > place_holder_type
- typedef variable< Tsor > variable_type
- typedef variable_state< Tsor > variable_state_type

Public Member Functions

- session ()
- session (session const &)=delete
- session (session &&)=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_

6.24.1 Member Typedef Documentation

6.24.1.1 place_holder_type

```
template<Tensor Tsor>
typedef place_holder<Tsor> ceras::session< Tsor >::place_holder_type
```

6.24.1.2 variable_state_type

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

6.24.1.3 variable_type

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

6.24.2 Constructor & Destructor Documentation

6.24.2.1 session() [1/3]

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

6.24.2.2 session() [2/3]

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

6.24.2.3 session() [3/3]

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

6.24.2.4 ~session()

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

6.24.3 Member Function Documentation

6.24.3.1 bind()

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

6.24.3.2 deserialize()

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

6.24.3.3 operator=() [1/2]

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

6.24.3.4 operator=() [2/2]

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

6.24.3.5 rebind()

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

6.24.3.6 remember()

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

6.24.3.7 restore()

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

6.24.3.8 run()

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

6.24.3.9 save()

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

6.24.3.10 serialize()

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

6.24.3.11 tap()

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

6.24.4 Member Data Documentation

6.24.4.1 place_holders_

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

6.24.4.2 variables_

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

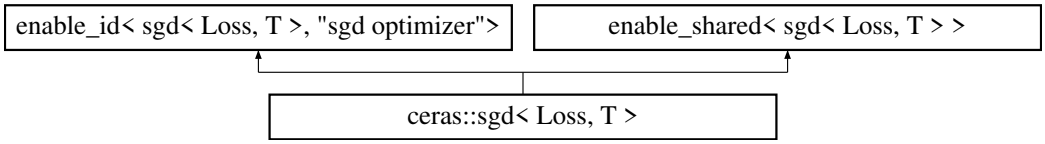
The documentation for this struct was generated from the following file:

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

6.25 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_

6.25.1 Member Typedef Documentation

6.25.1.1 tensor_type

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

6.25.2 Constructor & Destructor Documentation

6.25.2.1 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]
```

6.25.3 Member Function Documentation

6.25.3.1 forward()

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

6.25.4 Member Data Documentation

6.25.4.1 decay_

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

6.25.4.2 iterations_

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

6.25.4.3 learning_rate_

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

6.25.4.4 loss_

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

6.25.4.5 momentum_

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

6.25.4.6 nesterov_

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

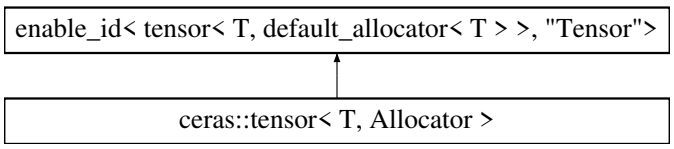
The documentation for this struct was generated from the following file:

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

6.26 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

Public Attributes

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

6.26.1 Member Typedef Documentation

6.26.1.1 allocator

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

6.26.1.2 self_type

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

6.26.1.3 shared_vector

```
template<typename T , typename Allocator = default_allocator<T>>
typedef std::shared_ptr<vector_type> ceras::tensor< T, Allocator >::shared_vector
```

6.26.1.4 value_type

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

6.26.1.5 vector_type

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

6.26.2 Constructor & Destructor Documentation

6.26.2.1 tensor() [1/7]

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

6.26.2.2 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]
```

6.26.2.3 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]
```

6.26.2.4 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]
```

6.26.2.5 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]
```

6.26.2.6 tensor() [6/7]

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

6.26.2.7 tensor() [7/7]

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

6.26.3 Member Function Documentation

6.26.3.1 begin() [1/2]

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

6.26.3.2 begin() [2/2]

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

6.26.3.3 cbegin()

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

6.26.3.4 cend()

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

6.26.3.5 copy()

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

6.26.3.6 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]
```

6.26.3.7 data() [1/2]

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

6.26.3.8 data() [2/2]

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

6.26.3.9 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]
```

6.26.3.10 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]
```

6.26.3.11 empty()

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

6.26.3.12 end() [1/2]

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

6.26.3.13 end() [2/2]

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

6.26.3.14 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]
```

6.26.3.15 ndim()

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

6.26.3.16 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]
```

6.26.3.17 operator*=() [2/2]

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

6.26.3.18 operator+=() [1/2]

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

6.26.3.19 operator+=() [2/2]

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

6.26.3.20 operator-()

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

6.26.3.21 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]
```

6.26.3.22 operator-=() [2/2]

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

6.26.3.23 operator/=() [1/2]

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

6.26.3.24 operator/=() [2/2]

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

6.26.3.25 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]
```

6.26.3.26 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]
```

6.26.3.27 operator[]() [1/2]

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

6.26.3.28 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]
```

6.26.3.29 reset()

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

6.26.3.30 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]
```

6.26.3.31 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]
```

6.26.3.32 shape()

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

6.26.3.33 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]
```

6.26.3.34 size()

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

6.26.3.35 slice()

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

6.26.4 Member Data Documentation

6.26.4.1 memory_offset_

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

6.26.4.2 shape_

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

6.26.4.3 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

6.27 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())>

6.27.1 Member Typedef Documentation

6.27.1.1 op_type

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

6.27.1.2 tensor_type

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

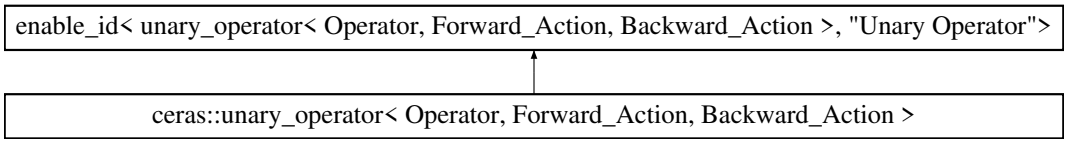
The documentation for this struct was generated from the following file:

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

6.28 ceras::unary_operator< Operator, Forward_Action, Backward_Action > Struct Template Reference

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

Inheritance diagram for ceras::unary_operator< Operator, Forward_Action, Backward_Action >:



Public Member Functions

- unary_operator (Operator const &op, Forward_Action const &forward_action, Backward_Action const &backward_action, 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_

6.28.1 Constructor & Destructor Documentation

6.28.1.1 unary_operator()

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
ceras::unary_operator< Operator, Forward_Action, Backward_Action >::unary_operator (
            Operator const & op,
            Forward_Action const & forward_action,
            Backward_Action const & backward_action,
            std::function< void()> const & reset_action )  [inline], [noexcept]
```

6.28.2 Member Function Documentation

6.28.2.1 backward()

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
void ceras::unary_operator< Operator, Forward_Action, Backward_Action >::backward (
            tensor_type const & grad )  [inline]
```

6.28.2.2 forward()

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

6.28.2.3 reset_states()

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

6.28.3 Member Data Documentation

6.28.3.1 backward_action_

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
Backward_Action ceras::unary_operator< Operator, Forward_Action, Backward_Action >::backward↩
_action_
```

6.28.3.2 forward_action_

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

6.28.3.3 input_data_

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

6.28.3.4 op_

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

6.28.3.5 output_data_

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

6.28.3.6 reset_action_

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

6.28.3.7 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

6.29 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_

6.29.1 Member Typedef Documentation

6.29.1.1 value_type

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

6.29.2 Constructor & Destructor Documentation

6.29.2.1 value() [1/4]

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

6.29.2.2 value() [2/4]

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

6.29.2.3 value() [3/4]

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

6.29.2.4 value() [4/4]

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

6.29.3 Member Function Documentation

6.29.3.1 backward()

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

6.29.3.2 forward()

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

6.29.3.3 operator=() [1/2]

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

6.29.3.4 operator=() [2/2]

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

6.29.4 Member Data Documentation

6.29.4.1 data_

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

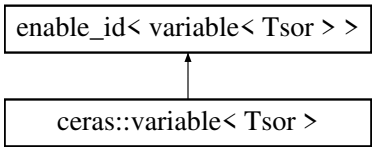
The documentation for this struct was generated from the following file:

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

6.30 ceras::variable< Tsor > Struct Template Reference

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

Inheritance diagram for ceras::variable< Tsor >:



Public Types

- typedef Tsor tensor_type

Public Member Functions

- variable (Tsor 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
- Tsor const forward () const
- void backward (auto const &grad)
- std::vector< std::size_t > shape () const noexcept
- Tsor & data ()
- Tsor data () const
- Tsor & gradient ()
- Tsor gradient () const
- void reset ()
- void reset_states ()

Public Attributes

- std::shared_ptr< variable_state< Tsor > > state_
- bool trainable_
- bool stateful_

6.30.1 Member Typedef Documentation

6.30.1.1 tensor_type

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

6.30.2 Constructor & Destructor Documentation

6.30.2.1 variable() [1/4]

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

6.30.2.2 variable() [2/4]

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

6.30.2.3 variable() [3/4]

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

6.30.2.4 variable() [4/4]

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

6.30.3 Member Function Documentation

6.30.3.1 backward()

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

6.30.3.2 data() [1/2]

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

6.30.3.3 data() [2/2]

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

6.30.3.4 forward()

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

6.30.3.5 gradient() [1/2]

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

6.30.3.6 gradient() [2/2]

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

6.30.3.7 operator=() [1/2]

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

6.30.3.8 operator=() [2/2]

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

6.30.3.9 reset()

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

6.30.3.10 reset_states()

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

6.30.3.11 shape()

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

6.30.4 Member Data Documentation

6.30.4.1 state_

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

6.30.4.2 stateful_

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

6.30.4.3 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

6.31 ceras::variable_state< Tsor > Struct Template Reference

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

Public Attributes

- Tsor data_
- Tsor gradient_

6.31.1 Member Data Documentation

6.31.1.1 data_

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

6.31.1.2 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

6.32 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_

6.32.1 Constructor & Destructor Documentation

6.32.1.1 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]
```

6.32.1.2 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]
```

6.32.1.3 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]
```

6.32.2 Member Function Documentation

6.32.2.1 data() [1/2]

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

6.32.2.2 data() [2/2]

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

6.32.2.3 operator[]() [1/2]

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

6.32.2.4 operator[]() [2/2]

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

6.32.2.5 shape()

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

6.32.2.6 size()

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

6.32.3 Member Data Documentation

6.32.3.1 col_

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

6.32.3.2 data_

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

6.32.3.3 row_

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

6.32.3.4 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

6.33 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_

6.33.1 Constructor & Destructor Documentation

6.33.1.1 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]
```

6.33.2 Member Function Documentation

6.33.2.1 operator[]() [1/2]

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

6.33.2.2 operator[]() [2/2]

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

6.33.3 Member Data Documentation

6.33.3.1 channel_

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

6.33.3.2 col_

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

6.33.3.3 data_

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

6.33.3.4 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

6.34 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.

6.34.1 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].

6.34.2 Constructor & Destructor Documentation

6.34.2.1 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

data	The raw pointer to the start position of the 1-D array.
batch_size	The first dimension of the 4-D tensor, also for the batch size in the CNN layers.
row	The second dimension of the 4-D tensor, also for the row in the CNN layers.
col	The third dimension of the 4-D tensor, also for the column in the CNN layers.
channel	The last dimension of the 4-D tensor, also for the channel in the CNN layers.

6.34.3 Member Function Documentation

6.34.3.1 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

index	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];
```

6.34.3.2 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

index	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;
```

6.34.4 Member Data Documentation

6.34.4.1 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.

6.34.4.2 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.

6.34.4.3 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.

6.34.4.4 data_

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

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

6.34.4.5 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

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, Rhs_Operator, Forward_Action, Backward_Action >
- struct ceras::is_operator< T >
- struct ceras::is_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > >
- struct ceras::is_operator< unary_operator< Operator, Forward_Action, Backward_Action > >

Namespaces

- ceras

Functions

- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::plus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::operator+ (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  auto ceras::operator* (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::log (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::negative (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::elementwise_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::hadamard_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::sum_reduce (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::reduce_sum (Ex const &ex) noexcept

- template<Expression Ex>
  constexpr auto ceras::mean_reduce (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::reduce_mean (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::minus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::operator- (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::square (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::abs (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::exp (Ex const &ex) noexcept
- template<typename Float >
  requires std::floating_point< Float > constexpr auto ceras::clip (Float lower, Float upper=std::numeric_↩limits< Float >::max()) noexcept
- auto ceras::reshape (std::vector< unsigned long > const &new_shape, bool include_batch_flag=true) noexcept
- template<Expression Ex>
  constexpr auto ceras::flatten (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::identity (Ex const &ex) noexcept
- template<Expression Ex>
  auto ceras::transpose (Ex const &ex) noexcept
- auto ceras::img2col (unsigned long const row_kernel, unsigned long col_kernel=-1, unsigned long const row_padding=0, unsigned long col_padding=0, unsigned long const row_stride=1, unsigned long const col↩_stride=1, unsigned long const row_dilation=1, unsigned long const col_dilation=1) noexcept
- auto ceras::conv2d (unsigned long row_input, unsigned long col_input, unsigned long const row_stride=1, unsigned long const col_stride=1, unsigned long const row_dilation=1, unsigned long const col_dilation=1, std::string const &padding="valid") noexcept
- template<typename T >
  requires std::floating_point< T > auto ceras::drop_out (T const factor) noexcept
- auto ceras::max_pooling_2d (unsigned long stride) noexcept
- auto ceras::average_pooling_2d (unsigned long stride) noexcept
- auto ceras::up_sampling_2d (unsigned long stride) noexcept
- template<typename T = double>
  requires std::floating_point< T > auto ceras::normalization_batch (T const momentum=0.98) noexcept
- template<typename T >
  requires std::floating_point< T > auto ceras::batch_normalization (T const momentum=0.98) noexcept
- template<typename T = double>
  requires std::floating_point< T > auto ceras::normalization_instance (T const momentum=0.98) noexcept
- template<typename T >
  requires std::floating_point< T > auto ceras::instance_normalization (T const momentum=0.98) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::concatenate (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- auto ceras::concatenate (unsigned long axe=-1)
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::concat (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- auto ceras::concat (unsigned long axe=-1)
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::maximum (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<typename T >
  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_operator_v = is_operator<T>::value
- template<typename T >
  concept ceras::Operator = is_operator_v<T>
- template<typename T >
  concept ceras::Expression = Operator<T> || Variable<T> || Place_Holder<T> || Constant<T> || Value<T>

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

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 >

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< Tsor >
- struct ceras::variable< Tsor >
- struct ceras::is_variable< T >
- struct ceras::is_variable< variable< Tsor > >

Namespaces

- ceras

Functions

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

Variables

- template<class T >
  constexpr bool ceras::is_variable_v = is_variable<T>::value
- template<typename T >
  concept ceras::Variable = is_variable_v<T>

Index

/data/structured_folders/workspace/github.repo/ceras/include/activation.hpp, 107
/data/structured_folders/workspace/github.repo/ceras/include/ceras.hpp, 108
/data/structured_folders/workspace/github.repo/ceras/include/configs.hpp, 108
/data/structured_folders/workspace/github.repo/ceras/include/constant.hpp, 109
/data/structured_folders/workspace/github.repo/ceras/include/includes.hpp, 109
/data/structured_folders/workspace/github.repo/ceras/include/kernels.hpp, 111
/data/structured_folders/workspace/github.repo/ceras/include/layer.hpp, 111
/data/structured_folders/workspace/github.repo/ceras/include/loss.hpp, 111
/data/structured_folders/workspace/github.repo/ceras/include/model.hpp, 112
/data/structured_folders/workspace/github.repo/ceras/include/operation.hpp, 113
/data/structured_folders/workspace/github.repo/ceras/include/optimizer.hpp, 115
/data/structured_folders/workspace/github.repo/ceras/include/place_holder.hpp, 116
/data/structured_folders/workspace/github.repo/ceras/include/recurrent_operation.hpp, 116
/data/structured_folders/workspace/github.repo/ceras/include/session.hpp, 117
/data/structured_folders/workspace/github.repo/ceras/include/tensor.hpp, 117
/data/structured_folders/workspace/github.repo/ceras/include/value.hpp, 121
/data/structured_folders/workspace/github.repo/ceras/include/variable.hpp, 122
__version__
    ceras, 40
~session
    ceras::session< Tsor >, 72

abs
    ceras, 16
abs_loss
    ceras, 17
ada_delta
    ceras, 16
ada_grad
    ceras, 16
adadelta
    ceras::adadelta< Loss, T >, 46
adagrad
    ceras::adagrad< Loss, T >, 48
adam
    ceras::adam< Loss, T >, 50
add
    ceras, 17
allocator
    ceras::tensor< T, Allocator >, 79
amax
    ceras, 17
amin
    ceras, 17
amsgrad_
    ceras::adam< Loss, T >, 50
as_tensor
    ceras, 17
average_pooling_2d
    ceras, 17

backward
    ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 53
    ceras::constant< Tsor >, 55
    ceras::place_holder< Tsor >, 67
    ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 89
    ceras::value< T >, 92
    ceras::variable< Tsor >, 95
backward_action_
    ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 53
    ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 90
batch_normalization
    ceras, 18
batch_size_
    ceras::view_4d< T >, 106
BatchNormalization
    ceras, 18
begin
    ceras::tensor< T, Allocator >, 81
beta_1_
    ceras::adam< Loss, T >, 50
beta_2_
    ceras::adam< Loss, T >, 51
binary_operator
    ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 52
bind
    ceras::place_holder< Tsor >, 67
    ceras::session< Tsor >, 73

blas_mode
ceras, 40

cbegin
ceras::tensor< T, Allocator >, 81
cend
ceras::tensor< T, Allocator >, 81
ceras, 9
__version__, 40
abs, 16
abs_loss, 17
ada_delta, 16
ada_grad, 16
add, 17
amax, 17
amin, 17
as_tensor, 17
average_pooling_2d, 17
batch_normalization, 18
BatchNormalization, 18
blas_mode, 40
clip, 18
concat, 18
concatenate, 19
Constant, 40
Conv2D, 19
conv2d, 19
copy, 20
cross_entropy, 20
cross_entropy_loss, 20
cuda_gemm_threshold, 40
cuda_mode, 41
debug_mode, 41
deep_copy, 20
default_allocator, 16
Dense, 21
drop_out, 21
elementwise_divide, 21
elementwise_product, 21
elu, 21
empty, 22
eps, 41
exp, 22
exponential, 22
Expression, 41
flatten, 22
gelu, 22
gemm, 22, 23
gemm_cpu, 23
get_default_session, 23
glorot_uniform, 23
hadamard_product, 23, 24
hard_sigmoid, 24
has_inf, 24
has_nan, 24
hinge_loss, 24
identity, 24
img2col, 25
Input, 25
instance_normalization, 25
is_constant_v, 41
is_operator_v, 41
is_place_holder_v, 41
is_tensor_v, 42
is_valid, 25
is_value_v, 42
is_variable_v, 42
is_windows_platform, 42
leaky_relu, 25
learning_phase, 42
linspace, 25
load_tensor, 26
log, 26
lstm, 43
mae, 26
make_binary_operator, 43
make_unary_operator, 43
max, 26
max_pooling_2d, 27
maximum, 27
mean, 27
mean_absolute_error, 27
mean_reduce, 27
mean_squared_error, 28
mean_squared_logarithmic_error, 28
min, 28
minus, 28
mse, 29
multiply, 29
negative, 29
negative_relu, 29
norm, 29
normalization_batch, 30
normalization_instance, 30
ones, 30
ones_like, 30
Operator, 43
operator<<, 32
operator*, 30, 31
operator+, 31
operator-, 32
operator/, 32
operator==, 33
Place_Holder, 43
plus, 33
randn, 33
randn_like, 33
random, 33
random_generator, 43
random_like, 34
random_normal_like, 34
random_seed, 44
read_tensor, 34
reduce, 34
reduce_mean, 35
reduce_sum, 35
relu, 35

repeat, 35
repmat, 36
reshape, 36
rms_prop, 16
save_tensor, 36
selu, 36
sigmoid, 36
softmax, 37
softplus, 37
softsign, 37
square, 37
squared_loss, 37
squeeze, 38
standard_deviation, 38
sum, 38
sum_reduce, 38
tanh, 38
Tensor, 44
transpose, 39
truncated_normal, 39
up_sampling_2d, 39
update_cuda_gemm_threshold, 39
Value, 44
Variable, 44
variance, 39
version, 44
visible_device, 44
write_tensor, 39
zeros, 40
zeros_like, 40
ceras::adadelta< Loss, T >, 45
adadelta, 46
forward, 46
iterations_, 46
learning_rate_, 46
loss_, 46
rho_, 47
tensor_type, 45
ceras::adagrad< Loss, T >, 47
adagrad, 48
decay_, 48
forward, 48
iterations_, 48
learning_rate_, 48
loss_, 49
tensor_type, 47
ceras::adam< Loss, T >, 49
adam, 50
amsgrad_, 50
beta_1_, 50
beta_2_, 51
forward, 50
iterations_, 51
learning_rate_, 51
loss_, 51
tensor_type, 50
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 51
backward, 53
backward_action_, 53
binary_operator, 52
forward, 53
forward_action_, 53
lhs_input_data_, 53
lhs_op_, 54
output_data_, 54
reset_action_, 54
reset_states, 53
rhs_input_data_, 54
rhs_op_, 54
tensor_type, 52
ceras::constant< Tsor >, 55
backward, 55
constant, 55
data_, 56
forward, 55
shape, 56
ceras::gradient_descent< Loss, T >, 56
forward, 57
gradient_descent, 57
learning_rate_, 57
loss_, 58
momentum_, 58
tensor_type, 57
ceras::is_constant< constant< Tsor > >, 59
ceras::is_constant< T >, 58
ceras::is_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > >, 59
ceras::is_operator< T >, 59
ceras::is_operator< unary_operator< Operator, Forward_Action, Backward_Action > >, 60
ceras::is_place_holder< place_holder< Tsor > >, 60
ceras::is_place_holder< T >, 60
ceras::is_tensor< T >, 61
ceras::is_tensor< tensor< T, A > >, 61
ceras::is_value< T >, 61
ceras::is_value< value< T > >, 62
ceras::is_variable< T >, 62
ceras::is_variable< variable< Tsor > >, 62
ceras::model< Ex, Ph >, 63
expression_, 65
model, 63
operator(), 64
place_holders_, 65
predict, 64
ceras::place_holder< Tsor >, 65
backward, 67
bind, 67
forward, 67
operator=, 67
place_holder, 66
reset, 67
tensor_type, 66
ceras::place_holder_state< Tsor >, 68
data_, 68

shape_hint_, 68
ceras::rmsprop< Loss, T >, 68
decay_, 70
forward, 69
iterations_, 70
learning_rate_, 70
loss_, 70
rho_, 70
rmsprop, 69
tensor_type, 69
ceras::session< Tsor >, 71
~session, 72
bind, 73
deserialize, 73
operator=, 73
place_holder_type, 71
place_holders_, 75
rebind, 73
remember, 73
restore, 74
run, 74
save, 74
serialize, 74
session, 72
tap, 74
variable_state_type, 71
variable_type, 72
variables_, 75
ceras::sgd< Loss, T >, 75
decay_, 76
forward, 76
iterations_, 77
learning_rate_, 77
loss_, 77
momentum_, 77
nesterov_, 77
sgd, 76
tensor_type, 76
ceras::tensor< T, Allocator >, 78
allocator, 79
begin, 81
cbegin, 81
cend, 81
copy, 82
creep_to, 82
data, 82
deep_copy, 82
empty, 83
end, 83
map, 83
memory_offset_, 87
ndim, 83
operator*=, 83, 84
operator+=, 84
operator-, 84
operator-=, 84
operator/=, 85
operator=, 85
operator[], 85
reset, 86
reshape, 86
resize, 86
self_type, 79
shape, 86
shape_, 87
shared_vector, 79
shrink_to, 86
size, 86
slice, 87
tensor, 80, 81
value_type, 79
vector_, 87
vector_type, 80
ceras::tensor_deduction< L, R >, 87
op_type, 88
tensor_type, 88
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 88
backward, 89
backward_action_, 90
forward, 89
forward_action_, 90
input_data_, 90
op_, 90
output_data_, 90
reset_action_, 90
reset_states, 89
tensor_type, 90
unary_operator, 89
ceras::value< T >, 91
backward, 92
data_, 93
forward, 92
operator=, 93
value, 92
value_type, 91
ceras::variable< Tsor >, 93
backward, 95
data, 95
forward, 95
gradient, 96
operator=, 96
reset, 96
reset_states, 96
shape, 96
state_, 97
stateful_, 97
tensor_type, 94
trainable_, 97
variable, 94, 95
ceras::variable_state< Tsor >, 97
data_, 97
gradient_, 98
ceras::view_2d< T >, 98
col_, 100
data, 99

data_, 100
operator[], 99, 100
row_, 100
shape, 100
size, 100
transposed_, 101
view_2d, 98, 99
ceras::view_3d< T >, 101
channel_, 102
col_, 102
data_, 102
operator[], 102
row_, 102
view_3d, 101
ceras::view_4d< T >, 103
batch_size_, 106
channel_, 106
col_, 106
data_, 106
operator[], 105
row_, 106
view_4d, 103
channel_
ceras::view_3d< T >, 102
ceras::view_4d< T >, 106
clip
ceras, 18
col_
ceras::view_2d< T >, 100
ceras::view_3d< T >, 102
ceras::view_4d< T >, 106
concat
ceras, 18
concatenate
ceras, 19
Constant
ceras, 40
constant
ceras::constant< Tsor >, 55
Conv2D
ceras, 19
conv2d
ceras, 19
copy
ceras, 20
ceras::tensor< T, Allocator >, 82
creep_to
ceras::tensor< T, Allocator >, 82
cross_entropy
ceras, 20
cross_entropy_loss
ceras, 20
cuda_gemm_threshold
ceras, 40
cuda_mode
ceras, 41

data
ceras::tensor< T, Allocator >, 82
ceras::variable< Tsor >, 95
ceras::view_2d< T >, 99
data_
ceras::constant< Tsor >, 56
ceras::place_holder_state< Tsor >, 68
ceras::value< T >, 93
ceras::variable_state< Tsor >, 97
ceras::view_2d< T >, 100
ceras::view_3d< T >, 102
ceras::view_4d< T >, 106
debug_mode
ceras, 41
decay_
ceras::adagrad< Loss, T >, 48
ceras::rmsprop< Loss, T >, 70
ceras::sgd< Loss, T >, 76
deep_copy
ceras, 20
ceras::tensor< T, Allocator >, 82
default_allocator
ceras, 16
Dense
ceras, 21
deserialize
ceras::session< Tsor >, 73
drop_out
ceras, 21

elementwise_divide
ceras, 21
elementwise_product
ceras, 21
elu
ceras, 21
empty
ceras, 22
ceras::tensor< T, Allocator >, 83
end
ceras::tensor< T, Allocator >, 83
eps
ceras, 41
exp
ceras, 22
exponential
ceras, 22
Expression
ceras, 41
expression_
ceras::model< Ex, Ph >, 65

flatten
ceras, 22
forward
ceras::adadelta< Loss, T >, 46
ceras::adagrad< Loss, T >, 48
ceras::adam< Loss, T >, 50
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 53
ceras::constant< Tsor >, 55

ceras::gradient_descent< Loss, T >, 57
ceras::place_holder< Tsor >, 67
ceras::rmsprop< Loss, T >, 69
ceras::sgd< Loss, T >, 76
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 89
ceras::value< T >, 92
ceras::variable< Tsor >, 95
forward_action_
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 53
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 90

gelu
ceras, 22
gemm
ceras, 22, 23
gemm_cpu
ceras, 23
get_default_session
ceras, 23
glorot_uniform
ceras, 23
gradient
ceras::variable< Tsor >, 96
gradient_
ceras::variable_state< Tsor >, 98
gradient_descent
ceras::gradient_descent< Loss, T >, 57

hadamard_product
ceras, 23, 24
hard_sigmoid
ceras, 24
has_inf
ceras, 24
has_nan
ceras, 24
hinge_loss
ceras, 24

identity
ceras, 24
img2col
ceras, 25
includes.hpp
STB_IMAGE_IMPLEMENTATION, 110
STB_IMAGE_RESIZE_IMPLEMENTATION, 110
STB_IMAGE_WRITE_IMPLEMENTATION, 110
Input
ceras, 25
input_data_
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 90
instance_normalization
ceras, 25
is_constant_v
ceras, 41
is_operator_v
ceras, 41
is_place_holder_v
ceras, 41
is_tensor_v
ceras, 42
is_valid
ceras, 25
is_value_v
ceras, 42
is_variable_v
ceras, 42
is_windows_platform
ceras, 42
iterations_
ceras::adadelta< Loss, T >, 46
ceras::adagrad< Loss, T >, 48
ceras::adam< Loss, T >, 51
ceras::rmsprop< Loss, T >, 70
ceras::sgd< Loss, T >, 77

leaky_relu
ceras, 25
learning_phase
ceras, 42
learning_rate_
ceras::adadelta< Loss, T >, 46
ceras::adagrad< Loss, T >, 48
ceras::adam< Loss, T >, 51
ceras::gradient_descent< Loss, T >, 57
ceras::rmsprop< Loss, T >, 70
ceras::sgd< Loss, T >, 77
lhs_input_data_
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 53
lhs_op_
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 54
linspace
ceras, 25
load_tensor
ceras, 26
log
ceras, 26
loss_
ceras::adadelta< Loss, T >, 46
ceras::adagrad< Loss, T >, 49
ceras::adam< Loss, T >, 51
ceras::gradient_descent< Loss, T >, 58
ceras::rmsprop< Loss, T >, 70
ceras::sgd< Loss, T >, 77
lstm
ceras, 43

mae
ceras, 26
make_binary_operator
ceras, 43
make_unary_operator

ceras, 43
map
ceras::tensor< T, Allocator >, 83
max
ceras, 26
max_pooling_2d
ceras, 27
maximum
ceras, 27
mean
ceras, 27
mean_absolute_error
ceras, 27
mean_reduce
ceras, 27
mean_squared_error
ceras, 28
mean_squared_logarithmic_error
ceras, 28
memory_offset_
ceras::tensor< T, Allocator >, 87
min
ceras, 28
minus
ceras, 28
model
ceras::model< Ex, Ph >, 63
momentum_
ceras::gradient_descent< Loss, T >, 58
ceras::sgd< Loss, T >, 77
mse
ceras, 29
multiply
ceras, 29

ndim
ceras::tensor< T, Allocator >, 83
negative
ceras, 29
negative_relu
ceras, 29
nesterov_
ceras::sgd< Loss, T >, 77
norm
ceras, 29
normalization_batch
ceras, 30
normalization_instance
ceras, 30

ones
ceras, 30
ones_like
ceras, 30
op_
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 90
op_type
ceras::tensor_deduction< L, R >, 88
Operator
ceras, 43
operator<<
ceras, 32
operator∗
ceras, 30, 31
operator∗=
ceras::tensor< T, Allocator >, 83, 84
operator()
ceras::model< Ex, Ph >, 64
operator+
ceras, 31
operator+=
ceras::tensor< T, Allocator >, 84
operator-
ceras, 32
ceras::tensor< T, Allocator >, 84
operator-=
ceras::tensor< T, Allocator >, 84
operator/
ceras, 32
operator/=
ceras::tensor< T, Allocator >, 85
operator=
ceras::place_holder< Tsor >, 67
ceras::session< Tsor >, 73
ceras::tensor< T, Allocator >, 85
ceras::value< T >, 93
ceras::variable< Tsor >, 96
operator==
ceras, 33
operator[]
ceras::tensor< T, Allocator >, 85
ceras::view_2d< T >, 99, 100
ceras::view_3d< T >, 102
ceras::view_4d< T >, 105
output_data_
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 54
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 90

Place_Holder
ceras, 43
place_holder
ceras::place_holder< Tsor >, 66
place_holder_type
ceras::session< Tsor >, 71
place_holders_
ceras::model< Ex, Ph >, 65
ceras::session< Tsor >, 75
plus
ceras, 33
predict
ceras::model< Ex, Ph >, 64

randn
ceras, 33
randn_like

ceras, 33
random
ceras, 33
random_generator
ceras, 43
random_like
ceras, 34
random_normal_like
ceras, 34
random_seed
ceras, 44
read_tensor
ceras, 34
rebind
ceras::session< Tsor >, 73
reduce
ceras, 34
reduce_mean
ceras, 35
reduce_sum
ceras, 35
relu
ceras, 35
remember
ceras::session< Tsor >, 73
repeat
ceras, 35
repmat
ceras, 36
reset
ceras::place_holder< Tsor >, 67
ceras::tensor< T, Allocator >, 86
ceras::variable< Tsor >, 96
reset_action_
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 54
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 90
reset_states
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 53
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 89
ceras::variable< Tsor >, 96
reshape
ceras, 36
ceras::tensor< T, Allocator >, 86
resize
ceras::tensor< T, Allocator >, 86
restore
ceras::session< Tsor >, 74
rho_
ceras::adadelta< Loss, T >, 47
ceras::rmsprop< Loss, T >, 70
rhs_input_data_
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 54
rhs_op_
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 54
rms_prop
ceras, 16
rmsprop
ceras::rmsprop< Loss, T >, 69
row_
ceras::view_2d< T >, 100
ceras::view_3d< T >, 102
ceras::view_4d< T >, 106
run
ceras::session< Tsor >, 74

save
ceras::session< Tsor >, 74
save_tensor
ceras, 36
self_type
ceras::tensor< T, Allocator >, 79
selu
ceras, 36
serialize
ceras::session< Tsor >, 74
session
ceras::session< Tsor >, 72
sgd
ceras::sgd< Loss, T >, 76
shape
ceras::constant< Tsor >, 56
ceras::tensor< T, Allocator >, 86
ceras::variable< Tsor >, 96
ceras::view_2d< T >, 100
shape_
ceras::tensor< T, Allocator >, 87
shape_hint_
ceras::place_holder_state< Tsor >, 68
shared_vector
ceras::tensor< T, Allocator >, 79
shrink_to
ceras::tensor< T, Allocator >, 86
sigmoid
ceras, 36
size
ceras::tensor< T, Allocator >, 86
ceras::view_2d< T >, 100
slice
ceras::tensor< T, Allocator >, 87
softmax
ceras, 37
softplus
ceras, 37
softsign
ceras, 37
square
ceras, 37
squared_loss
ceras, 37
squeeze
ceras, 38

standard_deviation
ceras, 38
state_
ceras::variable< Tsor >, 97
stateful_
ceras::variable< Tsor >, 97
STB_IMAGE_IMPLEMENTATION
includes.hpp, 110
STB_IMAGE_RESIZE_IMPLEMENTATION
includes.hpp, 110
STB_IMAGE_WRITE_IMPLEMENTATION
includes.hpp, 110
sum
ceras, 38
sum_reduce
ceras, 38

tanh
ceras, 38
tap
ceras::session< Tsor >, 74
Tensor
ceras, 44
tensor
ceras::tensor< T, Allocator >, 80, 81
tensor_type
ceras::adadelta< Loss, T >, 45
ceras::adagrad< Loss, T >, 47
ceras::adam< Loss, T >, 50
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, 52
ceras::gradient_descent< Loss, T >, 57
ceras::place_holder< Tsor >, 66
ceras::rmsprop< Loss, T >, 69
ceras::sgd< Loss, T >, 76
ceras::tensor_deduction< L, R >, 88
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 90
ceras::variable< Tsor >, 94
trainable_
ceras::variable< Tsor >, 97
transpose
ceras, 39
transposed_
ceras::view_2d< T >, 101
truncated_normal
ceras, 39

unary_operator
ceras::unary_operator< Operator, Forward_Action, Backward_Action >, 89
up_sampling_2d
ceras, 39
update_cuda_gemm_threshold
ceras, 39

Value
ceras, 44
value
ceras::value< T >, 92
value_type
ceras::tensor< T, Allocator >, 79
ceras::value< T >, 91
Variable
ceras, 44
variable
ceras::variable< Tsor >, 94, 95
variable_state_type
ceras::session< Tsor >, 71
variable_type
ceras::session< Tsor >, 72
variables_
ceras::session< Tsor >, 75
variance
ceras, 39
vector_
ceras::tensor< T, Allocator >, 87
vector_type
ceras::tensor< T, Allocator >, 80
version
ceras, 44
view_2d
ceras::view_2d< T >, 98, 99
view_3d
ceras::view_3d< T >, 101
view_4d
ceras::view_4d< T >, 103
visible_device
ceras, 44

write_tensor
ceras, 39

zeros
ceras, 40
zeros_like
ceras, 40