ceras

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 ceras Namespace Reference

Namespaces

- ceras_private
- dataset

Classes

- struct constant
- struct is_constant
- struct is_constant< constant< Tsor > >
- struct compiled_model
- struct model
- struct unary_operator
- struct binary_operator
- struct is_unary_operator
- struct is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >
- struct is_binary_operator
- struct is_binary_operator< binary_operator< Lhs_Operator, Rhs_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 tensor
- struct is_tensor
- struct is_tensor< tensor< T, A > >
- struct view_2d
- struct view_3d
- struct view_4d

- struct value
- struct is_value
- struct is_value< value< T > >
- struct tensor_deduction
- struct variable_state
- struct regularizer
- struct variable
- struct is_variable
- struct is_variable< variable< Tsor > >

Typedefs

- template<typename Loss , typename T >
  using ada_grad = adagrad< Loss, T >
- template<typename Loss , typename T >
  using rms_prop = rmsprop< Loss, T >
- template<typename Loss , typename T >
  using ada_delta = adadelta< Loss, T >
- template<typename T >
  using default_allocator = std::allocator< T >
- template<typename T >
  using matrix = view_2d< T >
- template<typename T >
  using cube = view_3d< T >
- template<typename T >
  using tesseract = view_4d< T >

Functions

- template<Expression Ex>
  constexpr auto softmax (Ex const &ex) noexcept
- template<Expression Ex>
  auto selu (Ex const &ex) noexcept
- template<Expression Ex>
  auto softplus (Ex const &ex) noexcept
- template<Expression Ex>
  auto softsign (Ex const &ex) noexcept
- template<Expression Ex>
  auto sigmoid (Ex const &ex) noexcept
- template<Expression Ex>
  auto tanh (Ex const &ex) noexcept
- template<Expression Ex>
  auto relu (Ex const &ex) noexcept
- template<typename T >
  requires std::floating_point< T > auto leaky_relu (T const factor) noexcept
- template<Expression Ex>
  auto negative_relu (Ex const &ex) noexcept
- template<typename T = float>
  requires std::floating_point< T > auto elu (T const alpha=1.0) noexcept
- template<Expression Ex>
  auto exponential (Ex const &ex) noexcept
- template<Expression Ex>
  auto hard_sigmoid (Ex const &ex) noexcept

- template<Expression Ex>
  auto gelu (Ex const &ex) noexcept
- auto Input ()
- auto Conv2D (unsigned long output_channels, std::vector< unsigned long > const &kernel_size, std↩ ::vector< unsigned long > const &input_shape, std::string const &padding="valid", std::vector< unsigned long > const &strides={1, 1}, std::vector< unsigned long > const &dilations={1, 1}, bool use_bias=true, float kernel_regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_↩ regularizer_l2=0.0f)

  2D convolution layer.
- auto Dense (unsigned long output_size, unsigned long input_size, bool use_bias=true, float kernel_↩ regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_regularizer↩ _l2=0.0f)

  Densly-connected layer.
- auto BatchNormalization (std::vector< unsigned long > const &shape, float threshold=0.95f, float kernel_↩ regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_regularizer_↩ l2=0.0f)

  Applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.
- auto BatchNormalization (float threshold, std::vector< unsigned long > const &shape, float kernel_↩ regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_regularizer↩ _l2=0.0f)
- auto Concatenate (unsigned long axis=-1) noexcept
- auto Add () noexcept
- auto Subtract () noexcept
- auto Multiply () noexcept
- template<Expression Ex>
  auto ReLU (Ex const &ex) noexcept
- auto Softmax () noexcept
- template<typename T = float>
  auto LeakyReLU (T const factor=0.2) noexcept
- template<typename T = float>
  auto ELU (T const factor=0.2) noexcept
- auto Reshape (std::vector< unsigned long > const &new_shape, bool include_batch_flag=true) noexcept
- auto Flatten () noexcept
- auto MaxPooling2D (unsigned long stride) noexcept
- auto UpSampling2D (unsigned long stride) noexcept
- template<typename T >
  auto Dropout (T factor) noexcept
- auto AveragePooling2D (unsigned long stride) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto mean_squared_logarithmic_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto squared_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto mean_squared_error (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noex- cept
- 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) noex- cept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto mae (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept

- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto cross_entropy (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto binary_cross_entropy_loss (Lhs_Expression const &ground_truth, Rhs_Expression const &prediction) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto cross_entropy_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto hinge_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  void make_trainable (Ex &ex, bool t)
- template<Expression Ex, Place_Holder Ph, Expression Ey>
  auto replace_placeholder_with_expression (Ex const &ex, Ph const &old_place_holder, Ey const &new_↩
  expression)
- template<typename Model , typename Optimizer , typename Loss >
  auto make_compiled_model (Model const &m, Loss const &l, Optimizer const &o)
- template<Expression Ex>
  std::string computation_graph (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto plus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto operator+ (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  auto operator∗ (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto log (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto negative (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto elementwise_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto elementwise_multiply (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto hadamard_product (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto sum_reduce (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto reduce_sum (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto mean_reduce (Ex const &ex) noexcept

  Computes the mean of elements across all dimensions of an expression.
- template<Expression Ex>
  constexpr auto reduce_mean (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto minus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto operator- (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto square (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto abs (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto exp (Ex const &ex) noexcept

- template<typename Float >
  requires std::floating_point< Float > constexpr auto clip (Float lower, Float upper=std::numeric_limits< Float >::max()) noexcept
- auto reshape (std::vector< unsigned long > const &new_shape, bool include_batch_flag=true) noexcept
- template<Expression Ex>
  constexpr auto flatten (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto identity (Ex const &ex) noexcept
- template<Expression Ex>
  auto transpose (Ex const &ex) noexcept
- auto img2col (unsigned long const row_kernel, unsigned long col_kernel=-1, unsigned long const row_↩ 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<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto concatenate (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- auto concatenate (unsigned long axe=-1)
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto concat (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- auto concat (unsigned long axe=-1)
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto maximum (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<typename T = float>
  requires std::floating_point< T > auto random_normal_like (T mean=0.0, T stddev=1.0) noexcept
- template<Expression Ex>
  auto ones_like (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto equal (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto sign (Ex const &ex) noexcept
- auto zero_padding_2d (std::vector< unsigned long > const &padding) noexcept

  Zero-padding layer for 2D input. The input should have 4-dimensions: `(batch_size, row, col, channel)`*. The output has 4-dimensions:* `(batch_size, new_row, new_col, channel)`*.*
- template<Place_Holder Ph>
  bool operator== (Ph const &lhs, Ph const &rhs)
- template<Place_Holder Ph>
  bool operator!= (Ph const &lhs, Ph const &rhs)
- template<Place_Holder Ph>
  bool operator< (Ph const &lhs, Ph const &rhs)
- template<Place_Holder Ph>
  bool operator> (Ph const &lhs, Ph const &rhs)
- template<Place_Holder Ph>
  bool operator<= (Ph const &lhs, Ph const &rhs)
- template<Place_Holder Ph>
  bool operator>= (Ph const &lhs, Ph const &rhs)

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

- template<Tensor Tsor>
  Tsor reduce_mean (Tsor const &tsor)
- template<Tensor Tsor>
  Tsor clip (Tsor &tsor, typename Tsor::value_type lower=0, typename Tsor::value_type upper=1)
- template<Tensor Tsor>
  Tsor squeeze (Tsor const &tsor)
- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > randn (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1})
- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > truncated_normal (std::vector< unsigned long > const &shape, T mean=T{0}, T stddev=T{1}, T lower=T{0}, T upper=T{1})
- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > random (std::vector< unsigned long > const &shape, T min=T{0}, T max=T{1})
- template<Tensor Tsor>
  Tsor random_like (Tsor const &tsor, typename Tsor::value_type min=0, typename Tsor::value_type max=1)
- template<Tensor Tsor>
  Tsor randn_like (Tsor const &tsor, typename Tsor::value_type mean=0, typename Tsor::value_type std-dev=1)
- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > glorot_uniform (std::initializer_list< unsigned long > shape)
- template<Tensor Tsor>
  Tsor deep_copy (Tsor const &tsor)
- template<Tensor Tsor>
  Tsor copy (Tsor const &tsor)
- template<Tensor Tsor>
  Tsor concatenate (Tsor const &lhs, Tsor const &rhs, unsigned long axis=0) noexcept
- template<Tensor Tsor>
  Tsor repmat (Tsor const &tsor, unsigned long row_rep, unsigned long col_rep)
- template<Tensor Tsor>
  constexpr bool empty (Tsor const &tsor) noexcept
- template<typename T , typename A = default_allocator<T>>
  constexpr tensor< T, A > zeros (std::vector< unsigned long > const &shape)
- template<Tensor Tsor>
  constexpr Tsor zeros_like (Tsor const &tsor)
- template<typename T , typename A = default_allocator<T>>
  constexpr tensor< T, A > ones (std::vector< unsigned long > const &shape)
- template<Tensor Tsor>
  constexpr Tsor ones_like (Tsor const &tsor)
- template<Tensor Tsor>
  auto max (Tsor const &tsor)
- template<Tensor Tsor>
  auto amax (Tsor const &tsor)
- template<Tensor Tsor>
  auto min (Tsor const &tsor)
- template<Tensor Tsor>
  auto amin (Tsor const &tsor)
- template<Tensor Tsor>
  auto sum (Tsor const &tsor)
- template<Tensor Tsor>
  auto mean (Tsor const &tsor)
- template<Tensor Tsor>
  auto norm (Tsor const &tsor)
- template<Tensor Tsor>
  Tsor abs (Tsor const &tsor)
- template<Tensor Tsor>
  Tsor softmax (Tsor const &tsor)

- template<Tensor Tsor>
  bool has_nan (Tsor const &tsor)
- template<Tensor Tsor>
  bool has_inf (Tsor const &tsor)
- template<Tensor Tsor>
  bool is_valid (Tsor const &tsor)
- template<Tensor Tsor, typename Function >
  Tsor reduce (Tsor const &ts, unsigned long axis, typename Tsor::value_type const &init, Function const &func, bool keepdims=false) noexcept
- template<Tensor Tsor>
  Tsor sum (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept
- template<Tensor Tsor>
  requires std::floating_point< typename Tsor::value_type > Tsor mean (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept
- template<Tensor Tsor>
  requires std::floating_point< typename Tsor::value_type > Tsor variance (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept
- template<Tensor Tsor>
  requires std::floating_point< typename Tsor::value_type > Tsor standard_deviation (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept
- template<Tensor Tsor>
  requires std::floating_point< typename Tsor::value_type > Tsor::value_type var (Tsor const &ts) noexcept
- template<Tensor Tsor>
  requires std::floating_point< typename Tsor::value_type > Tsor::value_type std (Tsor const &ts) noexcept
- template<Tensor Tsor>
  Tsor max (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept
- template<Tensor Tsor>
  Tsor min (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept
- template<typename T , typename A = default_allocator<T>>
  requires std::floating_point< T > tensor< T, A > linspace (T start, T stop, unsigned long num, bool endpoint=true) noexcept
- template<class _Tp , class _CharT , class _Traits , class _Alloc >
  std::basic_istream< _CharT, _Traits > & read_tensor (std::basic_istream< _CharT, _Traits > &__is, tensor< _Tp, _Alloc > &__x)
- template<class _Tp , class _CharT , class _Traits , class _Alloc >
  std::basic_ostream< _CharT, _Traits > & write_tensor (std::basic_ostream< _CharT, _Traits > &__os, tensor< _Tp, _Alloc > const &__x)
- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > load_tensor (std::string const &file_name)
- template<Tensor Tsor>
  void save_tensor (std::string const &file_name, Tsor const &tsor)
- template<Variable Var>
  bool operator== (Var const &lhs, Var const &rhs) noexcept

Variables

- template<class T >
  constexpr bool is_constant_v = is_constant<T>::value
- template<typename T >
  concept Constant = is_constant_v<T>
- auto MeanSquaredError

  Computes the mean of squares of errors between labels and predictions.
- auto MSE

  An alias name of function MeanSquaredError.
- auto MeanAbsoluteError

Computes the mean of absolute errors between labels and predictions.
- auto MAE

  An alias name of function MeanAbsoluteError.
- auto Hinge
- auto CategoricalCrossentropy
- auto CategoricalCrossEntropy
- auto BinaryCrossentropy
- auto BinaryCrossEntropy
- static constexpr auto make_unary_operator
- static constexpr auto make_binary_operator
- template<class T >
  constexpr bool is_unary_operator_v = is_unary_operator<T>::value
- template<typename T >
  concept Unary_Operator = is_unary_operator_v<T>

  A type that represents an unary operator.
- template<class T >
  constexpr bool is_binary_operator_v = is_binary_operator<T>::value
- template<typename T >
  concept Binary_Operator = is_binary_operator_v<T>

  A type that represents a binary operator.
- template<typename T >
  concept Operator = Unary_Operator<T> || Binary_Operator<T>

  A type that represents an unary or a binary operator.
- template<typename T >
  concept Expression = Operator<T> || Variable<T> || Place_Holder<T> || Constant<T> || Value<T>

  A type that represents a unary operator, a binary operator, a variable, a place_holder, a constant or a value.
- auto Adam
- auto SGD
- auto Adagrad
- auto RMSprop
- auto Adadelta
- template<class T >
  constexpr bool is_place_holder_v = is_place_holder<T>::value
- template<typename T >
  concept Place_Holder = is_place_holder_v<T>
- static unsigned long random_seed = std::chrono::system_clock::now().time_since_epoch().count()
- static std::mt19937 random_generator {random_seed}
- template<class T >
  constexpr bool is_tensor_v = is_tensor<T>::value
- template<typename T >
  concept Tensor = is_tensor_v<T>
- template<class T >
  constexpr bool is_value_v = is_value<T>::value
- template<typename T >
  concept Value = is_value_v<T>
- template<class T >
  constexpr bool is_variable_v = is_variable<T>::value
- template<typename T >
  concept Variable = is_variable_v<T>

5.1.1 Typedef Documentation

5.1.1.1 ada_delta

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

5.1.1.2 ada_grad

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

5.1.1.3 cube

```
template<typename T >
using ceras::cube = typedef view_3d<T>
```

5.1.1.4 default_allocator

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

5.1.1.5 matrix

```
template<typename T >
using ceras::matrix = typedef view_2d<T>
```

5.1.1.6 rms_prop

```
template<typename Loss , typename T >
using ceras::rms_prop = typedef rmsprop< Loss, T >
```

5.1.1.7 tesseract

```
template<typename T >
using ceras::tesseract = typedef view_4d<T>
```

5.1.2 Function Documentation

5.1.2.1 abs() [1/2]

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

5.1.2.2 abs() [2/2]

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

5.1.2.3 abs_loss()

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

5.1.2.4 Add()

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

Layer that adds two layers

Example usage:

```
auto input = Input(); // (16, )
auto x1 = Dense( 8, 16 )( input );
auto x2 = Dense( 8, 16 )( input );
auto x3 = Add()( x1, x2 ); // equivalent to 'x1 + x2'
auto m = model{ input, x3 };
```

5.1.2.5 add()

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

5.1.2.6 amax()

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

5.1.2.7 amin()

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

5.1.2.8 as_tensor()

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

5.1.2.9 average_pooling_2d()

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

5.1.2.10 AveragePooling2D()

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

Average pooling operation for spatial data.

5.1.2.11 batch_normalization()

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

5.1.2.12 BatchNormalization() [1/2]

```
auto ceras::BatchNormalization (
            float threshold,
            std::vector< unsigned long > const & shape,
            float kernel_regularizer_l1 = 0.0f,
            float kernel_regularizer_l2 = 0.0f,
            float bias_regularizer_l1 = 0.0f,
            float bias_regularizer_l2 = 0.0f )  [inline]
```

5.1.2.13 BatchNormalization() [2/2]

```
auto ceras::BatchNormalization (
            std::vector< unsigned long > const & shape,
            float threshold = 0.95f,
            float kernel_regularizer_l1 = 0.0f,
            float kernel_regularizer_l2 = 0.0f,
            float bias_regularizer_l1 = 0.0f,
            float bias_regularizer_l2 = 0.0f )  [inline]
```

Applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

Parameters

shape	Dimensionality of the input shape.
threshold	Momentum for the moving average.
kernel_regularizer↩_l1	L1 regularizer for the kernel. Defaults to `0.0f`.
kernel_regularizer↩_l2	L2 regularizer for the kernel. Defaults to `0.0f`.
bias_regularizer_l1	L1 regularizer for the bias vector. Defaults to `0.0f`.
bias_regularizer_l2	L2 regularizer for the bias vector. Defaults to `0.0f`.

Example code:
```
auto a = variable{ random<float>( {12, 34, 56, 78} ) };
auto b = BatchNormalization( {34, 56, 78}, 0.8f )( a ); // note the leading dimension of 'a' is intepretated
       as batch size, and only the rest 3 dimensions are required here.
```

5.1.2.14 binary_cross_entropy_loss()

```
template<Expression Lhs_Expression, Expression Rhs_Expression>
constexpr auto ceras::binary_cross_entropy_loss (
            Lhs_Expression const & ground_truth,
            Rhs_Expression const & prediction )  [constexpr], [noexcept]
```

5.1.2.15 clip() [1/2]

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

5.1.2.16 clip() [2/2]

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

5.1.2.17 computation_graph()

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

Generating the computation graph, in `graph description language`.

Parameters

ex	An expression.

Returns

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

5.1.2.18 concat() [1/2]

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

5.1.2.19 concat() [2/2]

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

5.1.2.20 concatenate() [1/3]

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

5.1.2.21 concatenate() [2/3]

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

5.1.2.22 concatenate() [3/3]

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

5.1.2.23 Concatenate()

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

Layer that concatenates two layers.

Parameters

axis	The concatenation axis. Default to the last channel.

Example usage:

```
auto l1 = variable{ tensor<float>{ {12, 11, 3} } };
auto l2 = variable{ tensor<float>{ {12, 11, 4} } };
auto l12 = Concatenate()( l1, l2 ); // should be of shape (12, 11, 7)
```

5.1.2.24 Conv2D()

```
auto ceras::Conv2D (
             unsigned long output_channels,
             std::vector< unsigned long > const & kernel_size,
             std::vector< unsigned long > const & input_shape,
             std::string const & padding = "valid",
             std::vector< unsigned long > const & strides = {1,1},
             std::vector< unsigned long > const & dilations = {1, 1},
             bool use_bias = true,
             float kernel_regularizer_l1 = 0.0f,
             float kernel_regularizer_l2 = 0.0f,
             float bias_regularizer_l1 = 0.0f,
             float bias_regularizer_l2 = 0.0f )  [inline]
```

2D convolution layer.

Parameters

output_channels	Dimensionality of the output space.
kernel_size	The height and width of the convolutional window.
input_shape	Dimensionality of the input shape.
padding	`valid` or `same`. `valid` suggests no padding. `same` suggests zero padding. Defaults to `valid`.
strides	The strides along the height and width direction. Defaults to `(1, 1)`.
dilations	The dialation along the height and width direction. Defaults to `(1, 1)`.
use_bias	Wether or not use a bias vector. Defaults to `true`.
kernel_regularizer↵ _l1	L1 regularizer for the kernel. Defaults to `0.0f`.
kernel_regularizer↵ _l2	L2 regularizer for the kernel. Defaults to `0.0f`.
bias_regularizer_l1	L1 regularizer for the bias vector. Defaults to `0.0f`.
bias_regularizer_l2	L2 regularizer for the bias vector. Defaults to `0.0f`.

Example code:
```
auto x = Input{};
auto y = Conv2D( 32, {3, 3}, {28, 28, 1}, "same" )( x );
auto z = Flatten()( y );
auto u = Dense( 10, 28*28*32 )( z );
auto m = model{ x, u };
```

5.1.2.25 conv2d()

```
auto ceras::conv2d (
            unsigned long row_input,
            unsigned long col_input,
            unsigned long const row_stride = 1,
            unsigned long const col_stride = 1,
            unsigned long const row_dilation = 1,
            unsigned long const col_dilation = 1,
            std::string const & padding = "valid" )  [inline], [noexcept]
```

5.1.2.26 copy()

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

5.1.2.27 cross_entropy()

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

5.1.2.28 cross_entropy_loss()

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

5.1.2.29 deep_copy()

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

5.1.2.30 Dense()

```
auto ceras::Dense (
             unsigned long output_size,
             unsigned long input_size,
             bool use_bias = true,
             float kernel_regularizer_l1 = 0.0f,
             float kernel_regularizer_l2 = 0.0f,
             float bias_regularizer_l1 = 0.0f,
             float bias_regularizer_l2 = 0.0f )  [inline]
```

Densly-connected layer.

Parameters

output_size	Dimensionality of output shape. The output shape is `(batch_size, output_size)`.
input_size	Dimensionality of input shape. The input shape is `(batch_size, input_size)`.
use_bias	Using a bias vector or not. Defaults to `true`.
kernel_regularizer←_l1	L1 regularizer for the kernel. Defaults to `0.0f`.
kernel_regularizer←_l2	L2 regularizer for the kernel. Defaults to `0.0f`.
bias_regularizer_l1	L1 regularizer for the bias vector. Defaults to `0.0f`.
bias_regularizer_l2	L2 regularizer for the bias vector. Defaults to `0.0f`.

Example code:
```
auto x = Input{};
auto y = Dense( 10, 28*28 )( x );
auto m = model{ x, y };
```

5.1.2.31 drop_out()

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

5.1.2.32 Dropout()

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

Applies Dropout to the input.

5.1.2.33 elementwise_divide()

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

5.1.2.34 elementwise_multiply()

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

5.1.2.35 elementwise_product() [1/2]

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

5.1.2.36 elementwise_product() [2/2]

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

5.1.2.37 elu()

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

5.1.2.38 ELU()

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

Exponential Linear Unit.

5.1.2.39 empty()

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

5.1.2.40 equal()

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

Returns the truth value of (lhs == rhs) element-wise. [+1 for true, 0 for false]

Parameters

lhs_ex	The first operator.
rhs_ex	The second operator.

Returns

An instance of a binary operator that evaluate the element-wise equality of two input operators.

Example code:
```
auto l = variable<tensor<float»{ /*...*/ };
auto r = place_holder<tensor<float»{};
auto eq = equal(l, r);
```

5.1.2.41 exp()

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

5.1.2.42 exponential()

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

5.1.2.43 Flatten()

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

Flattens the input. Does not affect the batch size.

5.1.2.44 flatten()

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

5.1.2.45 gelu()

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

5.1.2.46 gemm() [1/2]

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

5.1.2.47 gemm() [2/2]

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

5.1.2.48 gemm_cpu()

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

5.1.2.49 get_default_session()

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

5.1.2.50 glorot_uniform()

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

5.1.2.51 hadamard_product() [1/2]

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

5.1.2.52 hadamard_product() [2/2]

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

5.1.2.53 hard_sigmoid()

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

5.1.2.54 has_inf()

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

5.1.2.55 has_nan()

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

5.1.2.56 hinge_loss()

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

5.1.2.57 identity()

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

5.1.2.58 img2col()

```
auto ceras::img2col (
             unsigned long const row_kernel,
             unsigned long col_kernel = -1,
             unsigned long const row_padding = 0,
             unsigned long col_padding = 0,
             unsigned long const row_stride = 1,
             unsigned long const col_stride = 1,
             unsigned long const row_dilation = 1,
             unsigned long const col_dilation = 1 )  [inline], [noexcept]
```

5.1.2.59 Input()

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

5.1.2.60 is_valid()

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

5.1.2.61 leaky_relu()

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

5.1.2.62 LeakyReLU()

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

leaky relu activation function.

5.1.2.63 linspace()

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

5.1.2.64 load_tensor()

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

5.1.2.65 log()

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

5.1.2.66 mae()

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

5.1.2.67 make_compiled_model()

```
template<typename Model , typename Optimizer , typename Loss >
auto ceras::make_compiled_model (
            Model const & m,
            Loss const & l,
            Optimizer const & o )  [inline]
```

5.1.2.68 make_trainable()

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

Setting an expression's trainable flag

5.1.2.69 max() [1/2]

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

5.1.2.70 max() [2/2]

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

5.1.2.71 max_pooling_2d()

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

5.1.2.72 maximum()

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

5.1.2.73 MaxPooling2D()

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

Max pooling operation for 2D spatial data.

5.1.2.74 mean() [1/2]

```
template<Tensor Tsor>
requires std::floating_point<typename Tsor::value_type> Tsor ceras::mean (
            Tsor const & ts,
            unsigned long axis,
            bool keepdims = false )  [noexcept]
```

5.1.2.75 mean() [2/2]

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

5.1.2.76 mean_absolute_error()

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

5.1.2.77 mean_reduce()

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

Computes the mean of elements across all dimensions of an expression.

Parameters

ex	Incoming expression.

Example code:
```
auto va = place_holder<tensor<float»{};
auto vb = variable{ random<float>{ 3, 4} };
auto diff = mean_reduce( va, vb );
```

5.1.2.78 mean_squared_error()

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

5.1.2.79 mean_squared_logarithmic_error()

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

5.1.2.80 min() [1/2]

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

5.1.2.81 min() [2/2]

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

5.1.2.82 minus() [1/2]

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

5.1.2.83 minus() [2/2]

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

5.1.2.84 mse()

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

5.1.2.85 Multiply()

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

Layer that elementwise multiplies two layers

Example usage:
```
auto input = Input(); // (16, )
auto x1 = Dense( 8, 16 )( input );
auto x2 = Dense( 8, 16 )( input );
auto x3 = Multiply()( x1, x2 ); // equivalent to `elementwise_multiply(x1, x2)`
auto m = model{ input, x3 };
```

5.1.2.86 multiply() [1/2]

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

5.1.2.87 multiply() [2/2]

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

5.1.2.88 negative()

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

5.1.2.89 negative_relu()

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

5.1.2.90 norm()

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

5.1.2.91 normalization_batch()

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

5.1.2.92 ones()

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

5.1.2.93 ones_like() [1/2]

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

`ones_like` produces a tensor of the same shape as the input expression, but with every element to be 1.

Returns

An unary operator that takes an unary operator, and producing an output tensor Example Code:

```
auto va = variable{ ones<float>({3, 3, 3}) };
auto v_rand = ones_like( va ); // this expression will produces a tensor of shape (3, 3, 3), with every
 element to be 1.
```

5.1.2.94 ones_like() [2/2]

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

5.1.2.95 operator"!=()

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

5.1.2.96 operator*() [1/4]

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

5.1.2.97 operator*() [2/4]

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

5.1.2.98 operator*() [3/4]

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

5.1.2.99 operator*() [4/4]

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

5.1.2.100 operator+() [1/4]

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

5.1.2.101 operator+() [2/4]

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

5.1.2.102 operator+() [3/4]

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

5.1.2.103 operator+() [4/4]

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

5.1.2.104 operator-() [1/4]

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

5.1.2.105 operator-() [2/4]

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

5.1.2.106 operator-() [3/4]

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

5.1.2.107 operator-() [4/4]

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

5.1.2.108 operator/()

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

5.1.2.109 operator<()

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

5.1.2.110 operator<<()

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

5.1.2.111 operator<=()

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

5.1.2.112 operator==() [1/2]

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

5.1.2.113 operator==() [2/2]

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

5.1.2.114 operator>()

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

5.1.2.115 operator>=()

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

5.1.2.116 plus()

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

5.1.2.117 randn()

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

5.1.2.118 randn_like()

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

5.1.2.119 random()

```
template<typename T , typename A = default_allocator<T>>
tensor<T,A> ceras::random (
             std::vector< unsigned long > const & shape,
             T min = T{0},
             T max = T{1} )
```

5.1.2.120 random_like()

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

5.1.2.121 random_normal_like()

```
template<typename T = float>
requires std::floating_point<T> auto ceras::random_normal_like (
             T mean = 0.0,
             T stddev = 1.0 )  [inline], [noexcept]
```

`random_normal_like` produces random tensor from a normal distribution

Parameters

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)
```

5.1.2.122 read_tensor()

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

5.1.2.123 reduce()

```
template<Tensor Tsor, typename Function >
Tsor ceras::reduce (
            Tsor const & ts,
            unsigned long axis,
            typename Tsor::value_type const & init,
            Function const & func,
            bool keepdims = false )  [noexcept]
```

5.1.2.124 reduce_mean() [1/2]

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

5.1.2.125 reduce_mean() [2/2]

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

5.1.2.126 reduce_sum() [1/2]

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

5.1.2.127 reduce_sum() [2/2]

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

5.1.2.128 relu()

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

5.1.2.129 ReLU()

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

Rectified Linear Unit activation function.

5.1.2.130 repeat()

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

5.1.2.131 replace_placeholder_with_expression()

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

Replacing a place_holder with an expression.

Parameters

ex	Can be a unary operator, binary operator, variable, place_holder, a constant or a value
old_place_holder	An place holder in `ex`
new_expression	An expression that will replace `old_place_holder` in `ex`.

Returns

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

5.1.2.132 repmat()

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

5.1.2.133 Reshape()

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

Reshapes inputs into the given shape.

5.1.2.134 reshape() [1/2]

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

5.1.2.135 reshape() [2/2]

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

5.1.2.136 save_tensor()

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

5.1.2.137 selu()

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

5.1.2.138 sigmoid()

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

5.1.2.139 sign()

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

Returns the sign. [1 for positive, 0 for 0 and -1 for negative]

Parameters

ex	The input operator.

Returns

An instance of a unary_operator that evaluate the sign of the input operator.

Example code:

```
auto e = variable<tensor<float»{ /*...*/ };
auto si = sign(e);
```

5.1.2.140 Softmax()

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

Softmax activation function.

5.1.2.141 softmax() [1/2]

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

5.1.2.142 softmax() [2/2]

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

5.1.2.143 softplus()

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

5.1.2.144 softsign()

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

5.1.2.145 square()

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

Returns the square of the input

Parameters

ex	The input operator.

Returns

An instance of a unary_operator that evaluate the squared value of the input operator.

Example code:

```
auto e = variable<tensor<float>>{ /*...*/ };
auto square = square(e);
```

5.1.2.146 squared_loss()

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

5.1.2.147 squeeze()

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

5.1.2.148 standard_deviation()

```
template<Tensor Tsor>
requires std::floating_point<typename Tsor::value_type> Tsor ceras::standard_deviation (
            Tsor const & ts,
            unsigned long axis,
            bool keepdims = false )  [noexcept]
```

5.1.2.149 std()

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

5.1.2.150 Subtract()

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

Layer that subtracts two layers

Example usage:
```
auto input = Input(); // (16, )
auto x1 = Dense( 8, 16 )( input );
auto x2 = Dense( 8, 16 )( input );
auto x3 = Subtract()( x1, x2 ); // equivalent to 'x1 - x2'
auto m = model{ input, x3 };
```

5.1.2.151 sum() [1/2]

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

5.1.2.152 sum() [2/2]

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

5.1.2.153 sum_reduce()

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

5.1.2.154 tanh()

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

5.1.2.155 transpose()

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

5.1.2.156 truncated_normal()

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

5.1.2.157 up_sampling_2d()

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

5.1.2.158 update_cuda_gemm_threshold()

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

5.1.2.159 UpSampling2D()

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

Upsampling layer for 2D inputs.

5.1.2.160 var()

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

5.1.2.161 variance()

```
template<Tensor Tsor>
requires std::floating_point<typename Tsor::value_type> Tsor ceras::variance (
            Tsor const & ts,
            unsigned long axis,
            bool keepdims = false )  [noexcept]
```

5.1.2.162 write_tensor()

```
template<class _Tp , class _CharT , class _Traits , class _Alloc >
std::basic_ostream<_CharT, _Traits>& ceras::write_tensor (
            std::basic_ostream< _CharT, _Traits > & __os,
            tensor< _Tp, _Alloc > const & __x )
```

5.1.2.163 zero_padding_2d()

```
auto ceras::zero_padding_2d (
            std::vector< unsigned long > const & padding )  [inline], [noexcept]
```

Zero-padding layer for 2D input. The input should have 4-dimensions: `(batch_size, row, col, channel)`. The output has 4-dimensions: `(batch_size, new_row, new_col, channel)`.

Parameters

padding	If a single integer, then apply symmetric padding to height and width. If two integers, then first is for height and the second is for width. If four integers, then is intepreted as<tt>(top_pad, bottom_pad, left_pad, right_pad).

Example code:
```
auto a = variable{ random<float>( {16, 16, 3} );
auto b = zero_padding_2d( {8,} )( a ); // shape for b is (8+16+8, 8+16+8, 3)
auto c = zero_padding_2d( {8, 4} )( a ); // shape for c is (8+16+8, 4+16+4, 3)
auto d = zero_padding_2d( {8, 4, 2, 1} )( a ); // shape for d is (8+16+4, 2+16+1, 3)
```

5.1.2.164 zeros()

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

5.1.2.165 zeros_like()

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

5.1.3 Variable Documentation

5.1.3.1 Adadelta

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

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

5.1.3.2 Adagrad

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

Initial value:

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

5.1.3.3 Adam

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

Initial value:

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

5.1.3.4 Binary_Operator

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

A type that represents a binary operator.

@concept Binary_Operator<>

5.1.3.5 BinaryCrossentropy

```
auto ceras::BinaryCrossentropy [inline]
```

Initial value:

```
= []()
    {
        return []<Expression Ex >( Ex const& output )
        {
            return [=]<Place_Holder Ph>( Ph const& ground_truth )
            {
                return binary_cross_entropy_loss( ground_truth, output );
            };
        };
    }
```

5.1.3.6 BinaryCrossEntropy

```
auto ceras::BinaryCrossEntropy  [inline]
```

Initial value:

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

5.1.3.7 CategoricalCrossentropy

```
auto ceras::CategoricalCrossentropy  [inline]
```

Initial value:

```
= []()
    {
        return []<Expression Ex >( Ex const& output )
        {
            return [=]<Place_Holder Ph>( Ph const& ground_truth )
            {
                return cross_entropy_loss( ground_truth, output );
            };
        };
    }
```

5.1.3.8 CategoricalCrossEntropy

```
auto ceras::CategoricalCrossEntropy  [inline]
```

Initial value:

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

5.1.3.9 Constant

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

5.1.3.10 Expression

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

A type that represents a unary operator, a binary operator, a variable, a place_holder, a constant or a value.

@concept Expression<>

5.1.3.11 Hinge

```
auto ceras::Hinge [inline]
```

Initial value:

```
= []()
    {
        return []<Expression Ex >( Ex const& output )
        {
            return [=]<Place_Holder Ph>( Ph const& ground_truth )
            {
                return hinge_loss( ground_truth, output );
            };
        };
    }
```

5.1.3.12 is_binary_operator_v

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

If T is an instance of a binary_operator, the constant value equals to `true`. Otherwise this value is `false`.

5.1.3.13 is_constant_v

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

5.1.3.14 is_place_holder_v

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

5.1.3.15 is_tensor_v

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

5.1.3.16 is_unary_operator_v

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

If T is an instance of a unary_operator, the constant value equals to `true`. Otherwise this value is `false`.

5.1.3.17 is_value_v

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

5.1.3.18 is_variable_v

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

5.1.3.19 MAE

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

Initial value:

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

An alias name of function MeanAbsoluteError.

5.1.3.20 make_binary_operator

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

Initial value:

```
= []( auto const& binary_forward_action, auto const& binary_backward_action, std::string const&
       name="Anonymous Binary Operator" ) noexcept
    {
        return [&binary_forward_action, &binary_backward_action, &name]( auto const& lhs_op, auto const&
       rhs_op ) noexcept
        {
            auto ans = binary_operator{ lhs_op, rhs_op, binary_forward_action, binary_backward_action };
            ans.name_ = name;
            return ans;
        };
    }
```

5.1.3.21 make_unary_operator

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

Initial value:

```
= []( auto const& unary_forward_action, auto const& unary_backward_action, std::string const&
       name="Anonymous Unary Operator" ) noexcept
    {
        return [&unary_forward_action, &unary_backward_action, &name]( auto const& op ) noexcept
        {
            auto ans = unary_operator{ op, unary_forward_action, unary_backward_action };
            ans.name_ = name;
            return ans;
        };
    }
```

5.1.3.22 MeanAbsoluteError

```
auto ceras::MeanAbsoluteError [inline]
```

Initial value:

```
= []()
    {
        return []<Expression Ex >( Ex const& output )
        {
            return [=]<Place_Holder Ph>( Ph const& ground_truth )
            {
                return mean_absolute_error( ground_truth, output );
            };
        };
    }
```

Computes the mean of absolute errors between labels and predictions.

```
auto input = place_holder<tensor<float»{};
auto v = variable<tensor<float»{ ones<float>({12, 34}) };
auto output = input * v;
auto m = model{ input, output };
auto cm = m.compile( MeanAbsoluteError(), Adam(128/*batch size*/, 0.01f/*learning rate*/) );
```

see also mean_absolute_error

5.1.3.23 MeanSquaredError

```
auto ceras::MeanSquaredError [inline]
```

Initial value:

```
= []()
    {
        return []<Expression Ex >( Ex const& output )
        {
            return [=]<Place_Holder Ph>( Ph const& ground_truth )
            {
                return mean_squared_error( ground_truth, output );
            };
        };
    }
```

Computes the mean of squares of errors between labels and predictions.

```
auto input = place_holder<tensor<float»{};
auto v = variable<tensor<float»{ ones<float>({12, 34}) };
auto output = input * v;
auto m = model{ input, output };
auto cm = m.compile( MeanSquareError(), Adam(128/*batch size*/, 0.01f/*learning rate*/) );
```

see also mean_squared_error

5.1.3.24 MSE

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

Initial value:

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

An alias name of function MeanSquaredError.

5.1.3.25 Operator

```
template<typename T >
concept ceras::Operator = Unary_Operator<T> || Binary_Operator<T>
```

A type that represents an unary or a binary operator.

@concept Operator<>

5.1.3.26 Place_Holder

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

5.1.3.27 random_generator

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

5.1.3.28 random_seed

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

5.1.3.29 RMSprop

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

Initial value:

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

5.1.3.30 SGD

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

Initial value:

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

5.1.3.31 Tensor

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

5.1.3.32 Unary_Operator

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

A type that represents an unary operator.

@concept Unary_Operator<>

5.1.3.33 Value

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

5.1.3.34 Variable

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

5.2 ceras::ceras_private Namespace Reference

Classes

- struct session

5.3 ceras::dataset Namespace Reference

Namespaces

- fashion_mnist
- mnist

5.4 ceras::dataset::fashion_mnist Namespace Reference

Functions

- auto load_data (std::string const &path=std::string{"./dataset/fashion_mnist"})

5.4.1 Function Documentation

5.4.1.1 load_data()

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

Loads the fashion-MNIST dataset.

Parameters

path	Path where to cache the dataset locally. Default to "./dataset/fashion_mnist", should be updated if running the program somewhere else.

Returns

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

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

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

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

5.5 ceras::dataset::mnist Namespace Reference

Functions

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

5.5.1 Function Documentation

5.5.1.1 load_data()

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

Loads the MNIST dataset.

Parameters

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

Returns

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

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

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

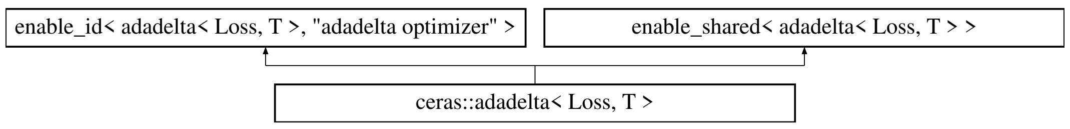
Chapter 6

Class Documentation

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

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

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


enable_id< adadelta< Loss, T >, "adadelta optimizer" >
enable_shared< adadelta< Loss, T > >
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_
```

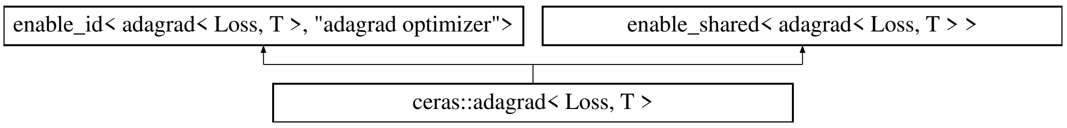
The documentation for this struct was generated from the following file:

- /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 >:


enable_id< adagrad< Loss, T >, "adagrad optimizer">
enable_shared< adagrad< Loss, T > >
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_
```

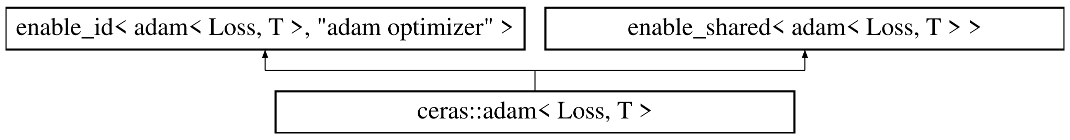
The documentation for this struct was generated from the following file:

- /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 >:


enable_id< adam< Loss, T >, "adam optimizer" >
enable_shared< adam< Loss, T > >
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_
```

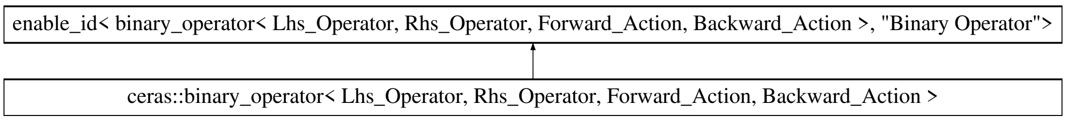
The documentation for this struct was generated from the following file:

- /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 >:


enable_id< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >, "Binary Operator">
ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >

Public Types

- typedef tensor_deduction< Lhs_Operator, Rhs_Operator >::tensor_type tensor_type

Public Member Functions

- binary_operator (Lhs_Operator const &lhs_op, Rhs_Operator const &rhs_op, Forward_Action const &forward_action, Backward_Action const &backward_action) noexcept
- auto forward ()
- void backward (tensor_type const &grad)

Public Attributes

- Lhs_Operator lhs_op_
- Rhs_Operator rhs_op_
- Forward_Action forward_action_
- Backward_Action backward_action_
- tensor_type lhs_input_data_
- tensor_type rhs_input_data_
- tensor_type output_data_

6.4.1 Member Typedef Documentation

6.4.1.1 tensor_type

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
typedef tensor_deduction<Lhs_Operator, Rhs_Operator>::tensor_type ceras::binary_operator<
Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action >::tensor_type
```

6.4.2 Constructor & Destructor Documentation

6.4.2.1 binary_operator()

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

6.4.3 Member Function Documentation

6.4.3.1 backward()

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

6.4.3.2 forward()

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

6.4.4 Member Data Documentation

6.4.4.1 backward_action_

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

6.4.4.2 forward_action_

```
template<typename Lhs_Operator , typename Rhs_Operator , typename Forward_Action , typename
Backward_Action >
Forward_Action ceras::binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_↩
Action >::forward_action_
```

6.4.4.3 lhs_input_data_

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

6.4.4.4 lhs_op_

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

6.4.4.5 output_data_

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

6.4.4.6 rhs_input_data_

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

6.4.4.7 rhs_op_

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

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

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

6.5 ceras::compiled_model< Model, Optimizer, Loss > Struct Template Reference

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

Public Types

- typedef Model::input_layer_type io_layer_type

Public Member Functions

- compiled_model (Model const &m, io_layer_type const &input_place_holder, io_layer_type const &ground↵_truth_place_holder, Loss const &loss, Optimizer const &optimizer)
- template<Tensor Tsor>
  auto evaluate (Tsor const &inputs, Tsor const &outputs, unsigned long batch_size=32)
- template<Tensor Tsor>
  auto fit (Tsor const &inputs, Tsor const &outputs, unsigned long batch_size, unsigned long epoch=1, int verbose=0, double validation_split=0.0)
- template<Tensor Tsor>
  auto train_on_batch (Tsor const &input, Tsor const &output)
- template<Tensor Tsor>
  auto predict (Tsor const &input_tensor)
- template<Expression Exp>
  auto operator() (Exp const &ex) const noexcept
- void trainable (bool t)

Public Attributes

- decltype(std::declval< Optimizer >()(std::declval< Loss & >())) typedef optimizer_type
- Model model_
- io_layer_type input_place_holder_
- io_layer_type ground_truth_place_holder_
- Loss loss_
- Optimizer optimizer_
- optimizer_type compiled_optimizer_

6.5.1 Member Typedef Documentation

6.5.1.1 io_layer_type

```
template<typename Model , typename Optimizer , typename Loss >
typedef Model::input_layer_type ceras::compiled_model< Model, Optimizer, Loss >::io_layer_type
```

6.5.2 Constructor & Destructor Documentation

6.5.2.1 compiled_model()

```
template<typename Model , typename Optimizer , typename Loss >
ceras::compiled_model< Model, Optimizer, Loss >::compiled_model (
            Model const & m,
            io_layer_type const & input_place_holder,
            io_layer_type const & ground_truth_place_holder,
            Loss const & loss,
            Optimizer const & optimizer )  [inline]
```

6.5.3 Member Function Documentation

6.5.3.1 evaluate()

```
template<typename Model , typename Optimizer , typename Loss >
template<Tensor Tsor>
auto ceras::compiled_model< Model, Optimizer, Loss >::evaluate (
             Tsor const & inputs,
             Tsor const & outputs,
             unsigned long batch_size = 32 )  [inline]
```

Calculate the loss for the model in test model.

Parameters

inputs	Input data. A tensor of shape (samples, input_shape).
outputs	Output data. A tensor of shape (samples, output_shape).
batch_size	Number of samples per batch of computation. Default to 32.

Returns

Test loss. A scalar.

6.5.3.2 fit()

```
template<typename Model , typename Optimizer , typename Loss >
template<Tensor Tsor>
auto ceras::compiled_model< Model, Optimizer, Loss >::fit (
             Tsor const & inputs,
             Tsor const & outputs,
             unsigned long batch_size,
             unsigned long epoch = 1,
             int verbose = 0,
             double validation_split = 0.0 )  [inline]
```

Train the model on the selected dataset for a fixed numbers of epoches.

Parameters

inputs	Input data. A tensor of shape (samples, input_shape).
outputs	Input data. A tensor of shape (samples, output_shape).
batch_size	Number of samples per gradient update. Should agree with the batch size in the optimizer.
epoch	Number of epoches to train the dataset.
verbose	Verbosity mode. 0 for slient. 1 for one line per epoch.
validation_split	Fraction of the training data that will be used for validation. A floating number in range [0, 1].

Returns

A tuple of two vectors. The first vector gives the historical errors on the training data. The second vector gives the historical errors on the validation data.

Example:

```
model m{ ... };
auto cm = m.compile( ... );
tensor<float> inputs, outputs;
//...
unsigned long batch_size = 32;
unsigned long epoch = 10;
int verbose = 1;
double validation_split = 0.2;
auto errors = cm.fit( inputs, outputs, batch_size, epoch, verbose, validation_split );
```

6.5.3.3 operator()()

```
template<typename Model , typename Optimizer , typename Loss >
template<Expression Exp>
auto ceras::compiled_model< Model, Optimizer, Loss >::operator() (
             Exp const & ex ) const  [inline], [noexcept]
```

6.5.3.4 predict()

```
template<typename Model , typename Optimizer , typename Loss >
template<Tensor Tsor>
auto ceras::compiled_model< Model, Optimizer, Loss >::predict (
             Tsor const & input_tensor )  [inline]
```

6.5.3.5 train_on_batch()

```
template<typename Model , typename Optimizer , typename Loss >
template<Tensor Tsor>
auto ceras::compiled_model< Model, Optimizer, Loss >::train_on_batch (
             Tsor const & input,
             Tsor const & output )  [inline]
```

Running a single updated on a single batch of data.

Parameters

input	The input data to train the model. A tensor of shape (batch_size, input_shape).
output	The output data to train the model. A tensor of shape (batch_size, output_shape).

Returns

Training loss. A scalar.

Example code:
```
auto m = model{ ... };
auto cm = m.compile( ... );
for ( auto idx : range( 1024 ) )
{
    auto x = ...; // get batch input
    auto y = ...; // get batch output
    cm.train_on_batch( x, y );
}
```

6.5.3.6 trainable()

```
template<typename Model , typename Optimizer , typename Loss >
void ceras::compiled_model< Model, Optimizer, Loss >::trainable (
               bool t )  [inline]
```

6.5.4 Member Data Documentation

6.5.4.1 compiled_optimizer_

```
template<typename Model , typename Optimizer , typename Loss >
optimizer_type ceras::compiled_model< Model, Optimizer, Loss >::compiled_optimizer_
```

6.5.4.2 ground_truth_place_holder_

```
template<typename Model , typename Optimizer , typename Loss >
io_layer_type ceras::compiled_model< Model, Optimizer, Loss >::ground_truth_place_holder_
```

6.5.4.3 input_place_holder_

```
template<typename Model , typename Optimizer , typename Loss >
io_layer_type ceras::compiled_model< Model, Optimizer, Loss >::input_place_holder_
```

6.5.4.4 loss_

```
template<typename Model , typename Optimizer , typename Loss >
Loss ceras::compiled_model< Model, Optimizer, Loss >::loss_
```

6.5.4.5 model_

```
template<typename Model , typename Optimizer , typename Loss >
Model ceras::compiled_model< Model, Optimizer, Loss >::model_
```

6.5.4.6 optimizer_

```
template<typename Model , typename Optimizer , typename Loss >
Optimizer ceras::compiled_model< Model, Optimizer, Loss >::optimizer_
```

6.5.4.7 optimizer_type

```
template<typename Model , typename Optimizer , typename Loss >
decltype(std::declval<Optimizer>()(std::declval<Loss&>())) typedef ceras::compiled_model<
Model, Optimizer, Loss >::optimizer_type
```

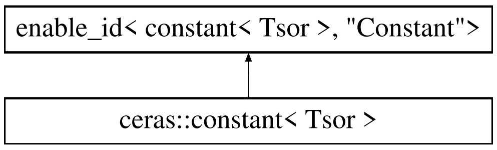
The documentation for this struct was generated from the following file:

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

6.6 ceras::constant< Tsor > Struct Template Reference

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

Inheritance diagram for ceras::constant< Tsor >:



Public Member Functions

- constant (Tsor const &data)
- void backward (auto) const
- Tsor forward () const
- auto shape () const

Public Attributes

- Tsor data_

6.6.1 Constructor & Destructor Documentation

6.6.1.1 constant()

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

6.6.2 Member Function Documentation

6.6.2.1 backward()

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

6.6.2.2 forward()

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

6.6.2.3 shape()

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

6.6.3 Member Data Documentation

6.6.3.1 data_

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

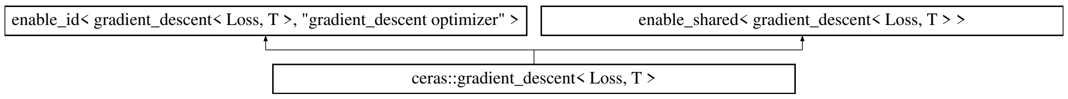
The documentation for this struct was generated from the following file:

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

6.7 ceras::gradient_descent< Loss, T > Struct Template Reference

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

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


enable_id< gradient_descent< Loss, T >, "gradient_descent optimizer" >
enable_shared< gradient_descent< Loss, T > >
ceras::gradient_descent< Loss, T >

Public Types

- typedef tensor< T > tensor_type

Public Member Functions

- gradient_descent (Loss &loss, std::size_t batch_size, T learning_rate=1.0e-3, T momentum=0.0) noexcept
- void forward ()

Public Attributes

- Loss & loss_
- T learning_rate_
- T momentum_

6.7.1 Member Typedef Documentation

6.7.1.1 tensor_type

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

6.7.2 Constructor & Destructor Documentation

6.7.2.1 gradient_descent()

```
template<typename Loss , typename T >
ceras::gradient_descent< Loss, T >::gradient_descent (
            Loss & loss,
            std::size_t batch_size,
            T learning_rate = 1.0e-3,
            T momentum = 0.0 )  [inline], [noexcept]
```

6.7.3 Member Function Documentation

6.7.3.1 forward()

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

6.7.4 Member Data Documentation

6.7.4.1 learning_rate_

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

6.7.4.2 loss_

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

6.7.4.3 momentum_

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

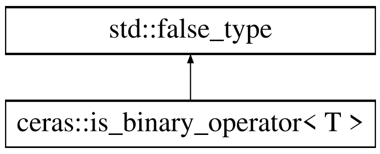
The documentation for this struct was generated from the following file:

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

6.8 ceras::is_binary_operator< T > Struct Template Reference

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

Inheritance diagram for ceras::is_binary_operator< T >:



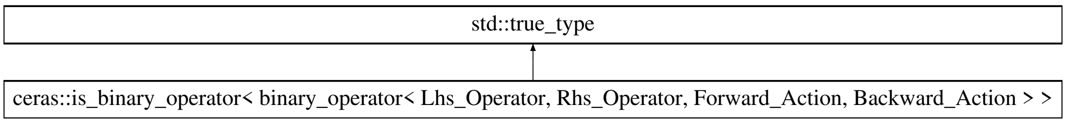
The documentation for this struct was generated from the following file:

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

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

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

Inheritance diagram for ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_↩ Action, Backward_Action > >:


std::true_type
ceras::is_binary_operator< binary_operator< Lhs_Operator, Rhs_Operator, Forward_Action, Backward_Action > >

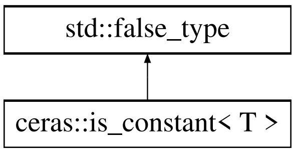
The documentation for this struct was generated from the following file:

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

6.10 ceras::is_constant< T > Struct Template Reference

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

Inheritance diagram for ceras::is_constant< T >:



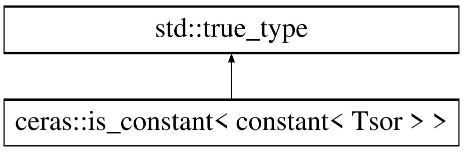
The documentation for this struct was generated from the following file:

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

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

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

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



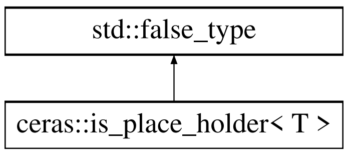
The documentation for this struct was generated from the following file:

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

6.12 ceras::is_place_holder< T > Struct Template Reference

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

Inheritance diagram for ceras::is_place_holder< T >:



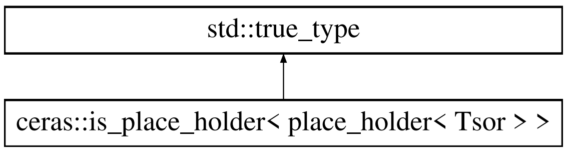
The documentation for this struct was generated from the following file:

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

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

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

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



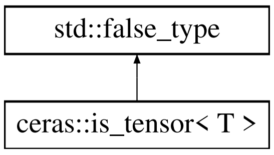
The documentation for this struct was generated from the following file:

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

6.14 ceras::is_tensor< T > Struct Template Reference

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

Inheritance diagram for ceras::is_tensor< T >:



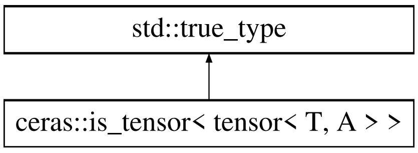
The documentation for this struct was generated from the following file:

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

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

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

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



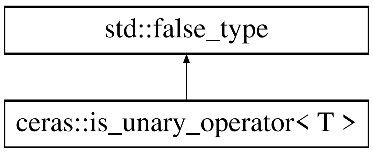
The documentation for this struct was generated from the following file:

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

6.16 ceras::is_unary_operator< T > Struct Template Reference

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

Inheritance diagram for ceras::is_unary_operator< T >:



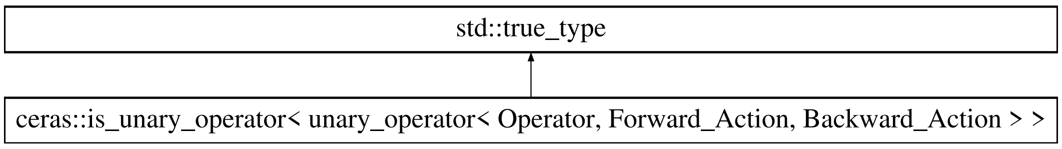
The documentation for this struct was generated from the following file:

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

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

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

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


std::true_type
ceras::is_unary_operator< unary_operator< Operator, Forward_Action, Backward_Action > >

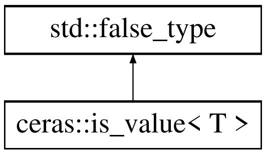
The documentation for this struct was generated from the following file:

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

6.18 ceras::is_value< T > Struct Template Reference

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

Inheritance diagram for ceras::is_value< T >:



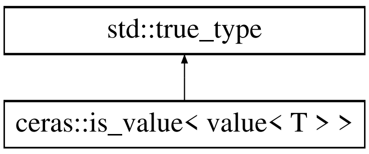
The documentation for this struct was generated from the following file:

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

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

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

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



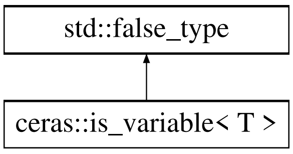
The documentation for this struct was generated from the following file:

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

6.20 ceras::is_variable< T > Struct Template Reference

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

Inheritance diagram for ceras::is_variable< T >:



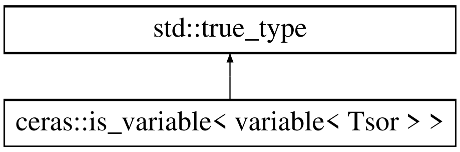
The documentation for this struct was generated from the following file:

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

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

`#include <variable.hpp>`

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



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

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

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

`#include <model.hpp>`

Public Types

- typedef Ph input_layer_type
- typedef Ex output_layer_type

Public Member Functions

- input_layer_type input () const noexcept
- output_layer_type output () const noexcept
- model (input_layer_type const &place_holder, output_layer_type const &expression)
- template<Tensor Tsor>
  auto predict (Tsor const &input_tensor)
- template<Expression Exp>
  auto operator() (Exp const &ex) const noexcept
- template<typename Loss , typename Optimizer >
  auto compile (Loss const &l, Optimizer const &o)
- void trainable (bool t)
- void save_weights (std::string const &file)
- void load_weights (std::string const &file)
- void summary (std::string const &file_name=std::string{}) const noexcept

Public Attributes

- output_layer_type expression_

  output layer of the model.
- input_layer_type place_holder_

6.22.1 Detailed Description

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

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

Template Parameters

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

6.22.2 Member Typedef Documentation

6.22.2.1 input_layer_type

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

6.22.2.2 output_layer_type

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

6.22.3 Constructor & Destructor Documentation

6.22.3.1 model()

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

Parameters

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 };
```

6.22.4 Member Function Documentation

6.22.4.1 compile()

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

Compile the model for training

Parameters

l	The loss to minimize.
o	The optimizer to do the optimization.

Returns

An instance of compiled_model.

Example useage:

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

6.22.4.2 input()

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

Returns the input layer of the model, which is a place_holder.

6.22.4.3 load_weights()

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

Loads all variables from a file

6.22.4.4 operator()()

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

Generating a new expression by using the current model.

Parameters

ex	An expression that represents the input to the model.

Returns

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

Example code

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

6.22.4.5 output()

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

Returns the output layer of the model.

6.22.4.6 predict()

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

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

Parameters

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)
```

6.22.4.7 save_weights()

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

Writes all variables to a file

6.22.4.8 summary()

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

Print the model summary to console or to a file.

Parameters

file_name	The file to save the summary. If empty, the summary will be printed to console. Empty by default.

6.22.4.9 trainable()

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

6.22.5 Member Data Documentation

6.22.5.1 expression_

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

output layer of the model.

6.22.5.2 place_holder_

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

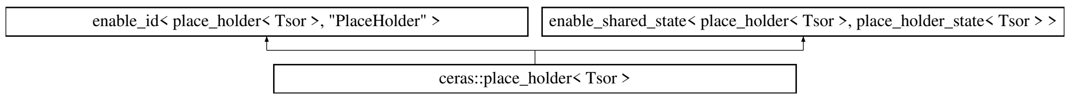
The documentation for this struct was generated from the following file:

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

6.23 ceras::place_holder< Tsor > Struct Template Reference

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

Inheritance diagram for ceras::place_holder< Tsor >:


enable_id< place_holder< Tsor >, "PlaceHolder" >
enable_shared_state< place_holder< Tsor >, place_holder_state< Tsor > >
ceras::place_holder< Tsor >

Public Types

- typedef Tsor tensor_type

Public Member Functions

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

6.23.1 Member Typedef Documentation

6.23.1.1 tensor_type

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

6.23.2 Constructor & Destructor Documentation

6.23.2.1 place_holder() [1/4]

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

6.23.2.2 place_holder() [2/4]

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

6.23.2.3 place_holder() [3/4]

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

6.23.2.4 place_holder() [4/4]

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

6.23.3 Member Function Documentation

6.23.3.1 backward()

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

6.23.3.2 bind()

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

6.23.3.3 forward()

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

6.23.3.4 operator=() [1/2]

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

6.23.3.5 operator=() [2/2]

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

6.23.3.6 reset()

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

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

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

6.24 ceras::place_holder_state< Tsor > Struct Template Reference

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

Public Attributes

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

6.24.1 Member Data Documentation

6.24.1.1 data_

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

6.24.1.2 shape_hint_

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

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

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

6.25 ceras::regularizer< Float > Struct Template Reference

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

Public Types

- typedef Float value_type

Public Member Functions

- constexpr regularizer (value_type l1, value_type l2, bool synchronized) noexcept

Public Attributes

- value_type l1_
- value_type l2_
- bool synchronized_

6.25.1 Member Typedef Documentation

6.25.1.1 value_type

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

6.25.2 Constructor & Destructor Documentation

6.25.2.1 regularizer()

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

6.25.3 Member Data Documentation

6.25.3.1 l1_

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

6.25.3.2 l2_

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

6.25.3.3 synchronized_

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

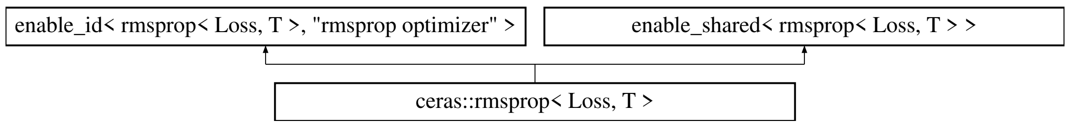
The documentation for this struct was generated from the following file:

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

6.26 ceras::rmsprop< Loss, T > Struct Template Reference

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

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


enable_id< rmsprop< Loss, T >, "rmsprop optimizer" >
enable_shared< rmsprop< Loss, T > >
ceras::rmsprop< Loss, T >

Public Types

- typedef tensor< T > tensor_type

Public Member Functions

- rmsprop (Loss &loss, std::size_t batch_size, T learning_rate=1.0e-1, T rho=0.9, T decay=0.0) noexcept
- void forward ()

Public Attributes

- Loss & loss_
- T learning_rate_
- T rho_
- T decay_
- unsigned long iterations_

6.26.1 Member Typedef Documentation

6.26.1.1 tensor_type

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

6.26.2 Constructor & Destructor Documentation

6.26.2.1 rmsprop()

```
template<typename Loss , typename T >
ceras::rmsprop< Loss, T >::rmsprop (
             Loss & loss,
             std::size_t batch_size,
             T learning_rate = 1.0e-1,
             T rho = 0.9,
             T decay = 0.0 )  [inline], [noexcept]
```

6.26.3 Member Function Documentation

6.26.3.1 forward()

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

6.26.4 Member Data Documentation

6.26.4.1 decay_

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

6.26.4.2 iterations_

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

6.26.4.3 learning_rate_

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

6.26.4.4 loss_

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

6.26.4.5 rho_

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

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

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

6.27 ceras::ceras_private::session< 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 &&)=default
- session & operator= (session const &)=delete
- session & operator= (session &&)=default
- 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::map< int, variable_type > variables_

6.27.1 Member Typedef Documentation

6.27.1.1 place_holder_type

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

6.27.1.2 variable_state_type

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

6.27.1.3 variable_type

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

6.27.2 Constructor & Destructor Documentation

6.27.2.1 session() [1/3]

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

6.27.2.2 session() [2/3]

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

6.27.2.3 session() [3/3]

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

6.27.2.4 ∼session()

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

6.27.3 Member Function Documentation

6.27.3.1 bind()

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

6.27.3.2 deserialize()

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

6.27.3.3 operator=() [1/2]

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

6.27.3.4 operator=() [2/2]

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

6.27.3.5 rebind()

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

6.27.3.6 remember()

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

6.27.3.7 restore()

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

6.27.3.8 run()

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

6.27.3.9 save()

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

6.27.3.10 serialize()

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

6.27.3.11 tap()

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

6.27.4 Member Data Documentation

6.27.4.1 place_holders_

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

6.27.4.2 variables_

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

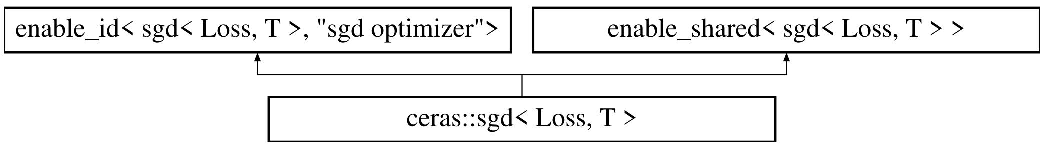
The documentation for this struct was generated from the following file:

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

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

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

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


enable_id< sgd< Loss, T >, "sgd optimizer">
enable_shared< sgd< Loss, T > >
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_

6.28.1 Member Typedef Documentation

6.28.1.1 tensor_type

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

6.28.2 Constructor & Destructor Documentation

6.28.2.1 sgd()

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

6.28.3 Member Function Documentation

6.28.3.1 forward()

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

6.28.4 Member Data Documentation

6.28.4.1 decay_

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

6.28.4.2 iterations_

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

6.28.4.3 learning_rate_

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

6.28.4.4 loss_

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

6.28.4.5 momentum_

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

6.28.4.6 nesterov_

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

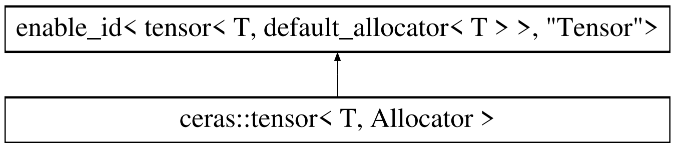
The documentation for this struct was generated from the following file:

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

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

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

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


enable_id< tensor< T, default_allocator< T > >, "Tensor">
ceras::tensor< T, Allocator >

Public Types

- typedef T value_type
- typedef Allocator allocator
- typedef std::vector< T, Allocator > vector_type
- typedef std::shared_ptr< vector_type > shared_vector
- typedef tensor self_type

Public Member Functions

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

- constexpr bool empty () const noexcept
- constexpr value_type ∗ data () noexcept
- constexpr const value_type ∗ data () const noexcept
- template<typename Function >
  constexpr self_type & map (Function const &f)
- constexpr self_type & operator+= (self_type const &other)
- constexpr self_type & operator+= (value_type x)
- constexpr self_type & operator-= (self_type const &other)
- constexpr self_type & operator-= (value_type x)
- constexpr self_type & operator∗= (self_type const &other)
- constexpr self_type & operator∗= (value_type x)
- constexpr self_type & operator/= (self_type const &other)
- constexpr self_type & operator/= (value_type x)
- constexpr self_type const operator- () const
- constexpr value_type as_scalar () const noexcept
- template<typename U >
  constexpr auto as_type () const noexcept

Public Attributes

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

6.29.1 Member Typedef Documentation

6.29.1.1 allocator

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

6.29.1.2 self_type

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

6.29.1.3 shared_vector

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

6.29.1.4 value_type

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

6.29.1.5 vector_type

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

6.29.2 Constructor & Destructor Documentation

6.29.2.1 tensor() [1/7]

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

6.29.2.2 tensor() [2/7]

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

6.29.2.3 tensor() [3/7]

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

6.29.2.4 tensor() [4/7]

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

6.29.2.5 tensor() [5/7]

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

6.29.2.6 tensor() [6/7]

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

6.29.2.7 tensor() [7/7]

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

6.29.3 Member Function Documentation

6.29.3.1 as_scalar()

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

6.29.3.2 as_type()

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

6.29.3.3 begin() [1/2]

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

6.29.3.4 begin() [2/2]

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

6.29.3.5 cbegin()

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

6.29.3.6 cend()

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

6.29.3.7 copy()

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

6.29.3.8 creep_to()

```
template<typename T , typename Allocator = default_allocator<T>>
constexpr self_type& ceras::tensor< T, Allocator >::creep_to (
             unsigned long new_memory_offset )  [inline], [constexpr]
```

6.29.3.9 data() [1/2]

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

6.29.3.10 data() [2/2]

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

6.29.3.11 deep_copy() [1/2]

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

6.29.3.12 deep_copy() [2/2]

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

6.29.3.13 empty()

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

6.29.3.14 end() [1/2]

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

6.29.3.15 end() [2/2]

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

6.29.3.16 map()

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

6.29.3.17 ndim()

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

6.29.3.18 operator*=() [1/2]

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

6.29.3.19 operator*=() [2/2]

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

6.29.3.20 operator+=() [1/2]

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

6.29.3.21 operator+=() [2/2]

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

6.29.3.22 operator-()

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

6.29.3.23 operator-=() [1/2]

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

6.29.3.24 operator-=() [2/2]

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

6.29.3.25 operator/=() [1/2]

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

6.29.3.26 operator/=() [2/2]

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

6.29.3.27 operator=() [1/2]

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

6.29.3.28 operator=() [2/2]

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

6.29.3.29 operator[]() [1/2]

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

6.29.3.30 operator[]() [2/2]

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

6.29.3.31 reset()

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

Resetting all elements in the tensor to a fixed value (default to 0), without change the shape.

Example code:
```
tensor<float> ts;
//...
ts.reset();
```

6.29.3.32 reshape()

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

6.29.3.33 resize()

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

6.29.3.34 shape()

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

6.29.3.35 shrink_to()

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

6.29.3.36 size()

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

6.29.3.37 slice()

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

6.29.4 Member Data Documentation

6.29.4.1 memory_offset_

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

6.29.4.2 shape_

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

6.29.4.3 vector_

```
template<typename T , typename Allocator = default_allocator<T>>
shared_vector ceras::tensor< T, Allocator >::vector_
```

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

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

6.30 ceras::tensor_deduction< L, R > Struct Template Reference

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

Public Types

- using op_type = std::conditional< is_value_v< L >, R, L >::type
- using tensor_type = std::remove_cv_t< decltype(std::declval< op_type >().forward())>

6.30.1 Member Typedef Documentation

6.30.1.1 op_type

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

6.30.1.2 tensor_type

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

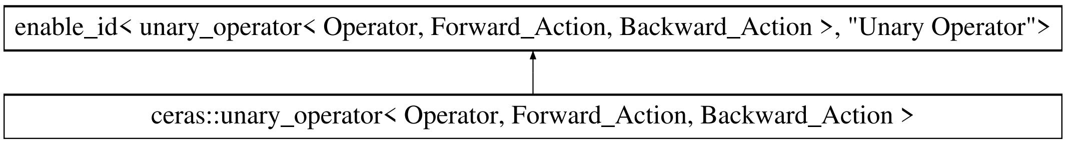
The documentation for this struct was generated from the following file:

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

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

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

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


enable_id< unary_operator< Operator, Forward_Action, Backward_Action >, "Unary Operator">
ceras::unary_operator< Operator, Forward_Action, Backward_Action >

Public Member Functions

- unary_operator (Operator const &op, Forward_Action const &forward_action, Backward_Action const &backward_action) noexcept
- auto forward ()
- void backward (tensor_type const &grad)

Public Attributes

- Operator op_
- Forward_Action forward_action_
- Backward_Action backward_action_
- decltype(std::declval< Forward_Action >()(std::declval< decltype(op_)>().forward())) typedef tensor_type
- tensor_type input_data_
- tensor_type output_data_

6.31.1 Constructor & Destructor Documentation

6.31.1.1 unary_operator()

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

6.31.2 Member Function Documentation

6.31.2.1 backward()

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

6.31.2.2 forward()

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

6.31.3 Member Data Documentation

6.31.3.1 backward_action_

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

6.31.3.2 forward_action_

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

6.31.3.3 input_data_

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

6.31.3.4 op_

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

6.31.3.5 output_data_

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

6.31.3.6 tensor_type

```
template<typename Operator , typename Forward_Action , typename Backward_Action >
decltype( std::declval<Forward_Action>()( std::declval<decltype(op_)>().forward() ) ) typedef
ceras::unary_operator< Operator, Forward_Action, Backward_Action >::tensor_type
```

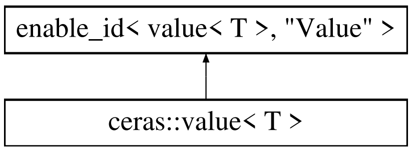
The documentation for this struct was generated from the following file:

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

6.32 ceras::value< T > Struct Template Reference

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

Inheritance diagram for ceras::value< T >:



Public Types

- typedef T value_type

Public Member Functions

- value ()=delete
- value (value_type v) noexcept
- value (value const &) noexcept=default
- value (value &&) noexcept=default
- value & operator= (value const &) noexcept=default
- value & operator= (value &&) noexcept=default
- void backward (auto) noexcept
- template<Tensor Tsor>
  Tsor const forward (Tsor const &refer) const

Public Attributes

- value_type data_

6.32.1 Member Typedef Documentation

6.32.1.1 value_type

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

6.32.2 Constructor & Destructor Documentation

6.32.2.1 value() [1/4]

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

6.32.2.2 value() [2/4]

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

6.32.2.3 value() [3/4]

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

6.32.2.4 value() [4/4]

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

6.32.3 Member Function Documentation

6.32.3.1 backward()

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

6.32.3.2 forward()

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

6.32.3.3 operator=() [1/2]

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

6.32.3.4 operator=() [2/2]

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

6.32.4 Member Data Documentation

6.32.4.1 data_

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

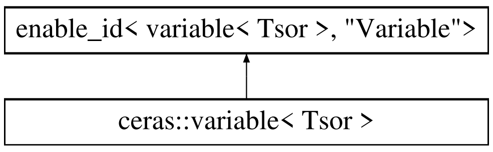
The documentation for this struct was generated from the following file:

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

6.33 ceras::variable< Tsor > Struct Template Reference

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

Inheritance diagram for ceras::variable< Tsor >:



Public Types

- typedef Tsor tensor_type
- typedef tensor_type::value_type value_type

Public Member Functions

- variable (tensor_type const &data, value_type l1=value_type{0}, value_type l2=value_type{0}, bool trainable=true)
- variable ()=delete
- variable (variable const &other)=default
- variable (variable &&)=default
- variable & operator= (variable &&)=default
- variable & operator= (variable const &other)=default
- tensor_type const forward () noexcept
- void backward (auto const &grad) noexcept
- std::vector< std::size_t > shape () const noexcept
- std::vector< tensor_type > & contexts ()
- std::vector< tensor_type > contexts () const
- tensor_type & data ()
- tensor_type data () const
- tensor_type & gradient ()
- tensor_type gradient () const
- void reset ()
- bool trainable () const noexcept
- void trainable (bool t)

Public Attributes

- std::shared_ptr< variable_state< tensor_type > > state_
- regularizer< value_type > regularizer_
- bool trainable_

6.33.1 Member Typedef Documentation

6.33.1.1 tensor_type

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

6.33.1.2 value_type

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

6.33.2 Constructor & Destructor Documentation

6.33.2.1 variable() [1/4]

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

6.33.2.2 variable() [2/4]

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

6.33.2.3 variable() [3/4]

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

6.33.2.4 variable() [4/4]

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

6.33.3 Member Function Documentation

6.33.3.1 backward()

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

6.33.3.2 contexts() [1/2]

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

6.33.3.3 contexts() [2/2]

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

6.33.3.4 data() [1/2]

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

6.33.3.5 data() [2/2]

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

6.33.3.6 forward()

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

6.33.3.7 gradient() [1/2]

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

6.33.3.8 gradient() [2/2]

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

6.33.3.9 operator=() [1/2]

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

6.33.3.10 operator=() [2/2]

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

6.33.3.11 reset()

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

6.33.3.12 shape()

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

6.33.3.13 trainable() [1/2]

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

6.33.3.14 trainable() [2/2]

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

6.33.4 Member Data Documentation

6.33.4.1 regularizer_

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

6.33.4.2 state_

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

6.33.4.3 trainable_

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

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

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

6.34 ceras::variable_state< Tsor > Struct Template Reference

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

Public Attributes

- Tsor data_
- Tsor gradient_
- std::vector< Tsor > contexts_

6.34.1 Member Data Documentation

6.34.1.1 contexts_

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

6.34.1.2 data_

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

6.34.1.3 gradient_

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

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

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

6.35 ceras::view_2d< T > Struct Template Reference

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

Public Member Functions

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

Public Attributes

- T ∗ data_
- unsigned long row_
- unsigned long col_
- bool transposed_

6.35.1 Constructor & Destructor Documentation

6.35.1.1 view_2d() [1/3]

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

6.35.1.2 view_2d() [2/3]

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

6.35.1.3 view_2d() [3/3]

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

6.35.2 Member Function Documentation

6.35.2.1 data() [1/2]

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

6.35.2.2 data() [2/2]

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

6.35.2.3 operator[]() [1/2]

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

6.35.2.4 operator[]() [2/2]

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

6.35.2.5 shape()

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

6.35.2.6 size()

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

6.35.3 Member Data Documentation

6.35.3.1 col_

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

6.35.3.2 data_

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

6.35.3.3 row_

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

6.35.3.4 transposed_

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

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

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

6.36 ceras::view_3d< T > Struct Template Reference

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

Public Member Functions

- constexpr view_3d (T ∗data, unsigned long row, unsigned long col, unsigned long channel) noexcept
- constexpr auto operator[ ] (unsigned long index) noexcept
- constexpr auto operator[ ] (unsigned long index) const noexcept

Public Attributes

- T ∗ data_
- unsigned long row_
- unsigned long col_
- unsigned long channel_

6.36.1 Constructor & Destructor Documentation

6.36.1.1 view_3d()

```
template<typename T >
constexpr ceras::view_3d< T >::view_3d (
             T * data,
             unsigned long row,
             unsigned long col,
             unsigned long channel )  [inline], [constexpr], [noexcept]
```

6.36.2 Member Function Documentation

6.36.2.1 operator[]() [1/2]

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

6.36.2.2 operator[]() [2/2]

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

6.36.3 Member Data Documentation

6.36.3.1 channel_

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

6.36.3.2 col_

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

6.36.3.3 data_

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

6.36.3.4 row_

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

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

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

6.37 ceras::view_4d< T > Struct Template Reference

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

Public Member Functions

- constexpr view_4d (T ∗data, unsigned long batch_size, unsigned long row, unsigned long col, unsigned long channel) noexcept
- constexpr auto operator[ ] (unsigned long index) noexcept
- constexpr auto operator[ ] (unsigned long index) const noexcept

Public Attributes

- T ∗ data_

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

  The batch size of the 4-D tensor, also the first dimension of the tensor.
- unsigned long row_

  The row of the 4-D tensor, also the second dimension of the tensor.
- unsigned long col_

  The column of the 4-D tensor, also the third dimension of the tensor.
- unsigned long channel_

  The channel of the 4-D tensor, also the last dimension of the tensor.

6.37.1 Detailed Description

template<typename T>
struct ceras::view_4d< T >

A class viewing a 1-D array as a 4-D tensor. This class is useful when treating an array as a typical 4-D tensor in a neural network, with a shape of [batch_size, row, column, channel].

6.37.2 Constructor & Destructor Documentation

6.37.2.1 view_4d()

```
template<typename T >
constexpr ceras::view_4d< T >::view_4d (
            T ∗ data,
            unsigned long batch_size,
            unsigned long row,
            unsigned long col,
            unsigned long channel )  [inline], [constexpr], [noexcept]
```

Constructor of view_4d

Parameters

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.

6.37.3 Member Function Documentation

6.37.3.1 operator[]() [1/2]

```
template<typename T >
constexpr auto ceras::view_4d< T >::operator[ ] (
              unsigned long index ) const  [inline], [constexpr], [noexcept]
```

Giving a view_3d interface for operator [].

Parameters

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];
```

6.37.3.2 operator[]() [2/2]

```
template<typename T >
constexpr auto ceras::view_4d< T >::operator[ ] (
              unsigned long index )  [inline], [constexpr], [noexcept]
```

Giving a view_3d interface for operator [].

Parameters

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;
```

6.37.4 Member Data Documentation

6.37.4.1 batch_size_

```
template<typename T >
unsigned long ceras::view_4d< T >::batch_size_
```

The batch size of the 4-D tensor, also the first dimension of the tensor.

6.37.4.2 channel_

```
template<typename T >
unsigned long ceras::view_4d< T >::channel_
```

The channel of the 4-D tensor, also the last dimension of the tensor.

6.37.4.3 col_

```
template<typename T >
unsigned long ceras::view_4d< T >::col_
```

The column of the 4-D tensor, also the third dimension of the tensor.

6.37.4.4 data_

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

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

6.37.4.5 row_

```
template<typename T >
unsigned long ceras::view_4d< T >::row_
```

The row of the 4-D tensor, also the second dimension of the tensor.

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

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

Chapter 7

File Documentation

7.1 /data/structured_↩ folders/workspace/github.repo/ceras/include/activation.hpp File Reference

```
#include "./operation.hpp"
#include "./tensor.hpp"
#include "./utils/range.hpp"
#include "./utils/better_assert.hpp"
#include "./utils/for_each.hpp"
#include "./utils/context_cast.hpp"
```

Namespaces

- ceras

Functions

- template< Expression Ex >
  constexpr auto ceras::softmax (Ex const &ex) noexcept
- template< Expression Ex >
  auto ceras::selu (Ex const &ex) noexcept
- template< Expression Ex >
  auto ceras::softplus (Ex const &ex) noexcept
- template< Expression Ex >
  auto ceras::softsign (Ex const &ex) noexcept
- template< Expression Ex >
  auto ceras::sigmoid (Ex const &ex) noexcept
- template< Expression Ex >
  auto ceras::tanh (Ex const &ex) noexcept
- template< Expression Ex >
  auto ceras::relu (Ex const &ex) noexcept
- template< typename T >
  requires std::floating_point< T > auto ceras::leaky_relu (T const factor) noexcept

- template<Expression Ex>
  auto ceras::negative_relu (Ex const &ex) noexcept
- template<typename T = float>
  requires std::floating_point< T > auto ceras::elu (T const alpha=1.0) noexcept
- template<Expression Ex>
  auto ceras::exponential (Ex const &ex) noexcept
- template<Expression Ex>
  auto ceras::hard_sigmoid (Ex const &ex) noexcept
- template<Expression Ex>
  auto ceras::gelu (Ex const &ex) noexcept

7.2 /data/structured_↩ folders/workspace/github.repo/ceras/include/ceras.hpp File Reference

```
#include "./config.hpp"
#include "./includes.hpp"
#include "./activation.hpp"
#include "./ceras.hpp"
#include "./loss.hpp"
#include "./operation.hpp"
#include "./optimizer.hpp"
#include "./place_holder.hpp"
#include "./session.hpp"
#include "./tensor.hpp"
#include "./variable.hpp"
#include "./constant.hpp"
#include "./layer.hpp"
#include "./model.hpp"
#include "./dataset.hpp"
```

7.3 /data/structured_↩ folders/workspace/github.repo/ceras/include/config.hpp File Reference

7.4 /data/structured_↩ folders/workspace/github.repo/ceras/include/constant.hpp File Reference

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

Classes

- struct ceras::constant< Tsor >
- struct ceras::is_constant< T >
- struct ceras::is_constant< constant< Tsor > >

Namespaces

- ceras

Variables

- template<class T >
  constexpr bool ceras::is_constant_v = is_constant<T>::value
- template<typename T >
  concept ceras::Constant = is_constant_v<T>

7.5 /data/structured_↩ folders/workspace/github.repo/ceras/include/dataset.hpp File Reference

```
#include "./tensor.hpp"
#include "./includes.hpp"
#include "./utils/better_assert.hpp"
#include "./utils/for_each.hpp"
```

Namespaces

- ceras
- ceras::dataset
- ceras::dataset::mnist
- ceras::dataset::fashion_mnist

Functions

- auto ceras::dataset::mnist::load_data (std::string const &path=std::string{"./dataset/mnist"})
- auto ceras::dataset::fashion_mnist::load_data (std::string const &path=std::string{"./dataset/fashion_mnist"})

7.6 /data/structured_↩ folders/workspace/github.repo/ceras/include/includes.hpp File Reference

```
#include "./config.hpp"
#include <algorithm>
#include <any>
#include <array>
#include <cassert>
#include <chrono>
#include <cmath>
#include <compare>
#include <concepts>
#include <cstdint>
#include <ctime>
#include <filesystem>
#include <fstream>
#include <functional>
#include <initializer_list>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <limits>
#include <map>
#include <memory>
#include <numeric>
#include <optional>
#include <ostream>
#include <random>
#include <ranges>
#include <regex>
#include <set>
#include <sstream>
#include <string>
#include <tuple>
#include <thread>
#include <type_traits>
#include <unordered_map>
#include <unordered_set>
#include <utility>
#include <vector>
#include "./utils/3rd_party/stb_image.h"
#include "./utils/3rd_party/stb_image_write.h"
#include "./utils/3rd_party/stb_image_resize.h"
#include "./utils/3rd_party/glob.hpp"
```

Macros

- #define STB_IMAGE_IMPLEMENTATION
- #define STB_IMAGE_WRITE_IMPLEMENTATION
- #define STB_IMAGE_RESIZE_IMPLEMENTATION

7.6.1 Macro Definition Documentation

7.6.1.1 STB_IMAGE_IMPLEMENTATION

```
#define STB_IMAGE_IMPLEMENTATION
```

7.6.1.2 STB_IMAGE_RESIZE_IMPLEMENTATION

```
#define STB_IMAGE_RESIZE_IMPLEMENTATION
```

7.6.1.3 STB_IMAGE_WRITE_IMPLEMENTATION

```
#define STB_IMAGE_WRITE_IMPLEMENTATION
```

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

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

Namespaces

- ceras

Functions

- auto ceras::Input ()
- auto ceras::Conv2D (unsigned long output_channels, std::vector< unsigned long > const &kernel_size, std↩ ::vector< unsigned long > const &input_shape, std::string const &padding="valid", std::vector< unsigned long > const &strides={1, 1}, std::vector< unsigned long > const &dilations={1, 1}, bool use_bias=true, float kernel_regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_↩ regularizer_l2=0.0f)

  2D convolution layer.
- auto ceras::Dense (unsigned long output_size, unsigned long input_size, bool use_bias=true, float kernel↩ _regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_regularizer_↩ l2=0.0f)

  Densly-connected layer.
- auto ceras::BatchNormalization (std::vector< unsigned long > const &shape, float threshold=0.95f, float kernel_regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_↩ regularizer_l2=0.0f)

Applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

- auto ceras::BatchNormalization (float threshold, std::vector< unsigned long > const &shape, float kernel↩ _regularizer_l1=0.0f, float kernel_regularizer_l2=0.0f, float bias_regularizer_l1=0.0f, float bias_regularizer_↩ l2=0.0f)
- auto ceras::Concatenate (unsigned long axis=-1) noexcept
- auto ceras::Add () noexcept
- auto ceras::Subtract () noexcept
- auto ceras::Multiply () noexcept
- template< Expression Ex>
  auto ceras::ReLU (Ex const &ex) noexcept
- auto ceras::Softmax () noexcept
- template< typename T = float>
  auto ceras::LeakyReLU (T const factor=0.2) noexcept
- template< typename T = float>
  auto ceras::ELU (T const factor=0.2) noexcept
- auto ceras::Reshape (std::vector< unsigned long > const &new_shape, bool include_batch_flag=true) noexcept
- auto ceras::Flatten () noexcept
- auto ceras::MaxPooling2D (unsigned long stride) noexcept
- auto ceras::UpSampling2D (unsigned long stride) noexcept
- template< typename T >
  auto ceras::Dropout (T factor) noexcept
- auto ceras::AveragePooling2D (unsigned long stride) noexcept

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

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

Namespaces

- ceras

Functions

- template< Expression Lhs_Expression, Expression 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::binary_cross_entropy_loss (Lhs_Expression const &ground_truth, Rhs_Expression const &prediction) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::cross_entropy_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::hinge_loss (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept

Variables

- auto ceras::MeanSquaredError

  Computes the mean of squares of errors between labels and predictions.
- auto ceras::MSE

  An alias name of function MeanSquaredError.
- auto ceras::MeanAbsoluteError

  Computes the mean of absolute errors between labels and predictions.
- auto ceras::MAE

  An alias name of function MeanAbsoluteError.
- auto ceras::Hinge
- auto ceras::CategoricalCrossentropy
- auto ceras::CategoricalCrossEntropy
- auto ceras::BinaryCrossentropy
- auto ceras::BinaryCrossEntropy

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

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

Classes

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

Namespaces

- ceras

Functions

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

7.10 /data/structured_↩ folders/workspace/github.repo/ceras/include/operation.hpp File Reference

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

Classes

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

Namespaces

- ceras

Functions

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

  Computes the mean of elements across all dimensions of an expression.
- template<Expression Ex>
  constexpr auto ceras::reduce_mean (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::minus (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::operator- (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::square (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::abs (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::exp (Ex const &ex) noexcept
- template<typename Float >
  requires std::floating_point< Float > constexpr auto ceras::clip (Float lower, Float upper=std::numeric_↩ limits< Float >::max()) noexcept
- auto ceras::reshape (std::vector< unsigned long > const &new_shape, bool include_batch_flag=true) noexcept
- template<Expression Ex>
  constexpr auto ceras::flatten (Ex const &ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::identity (Ex const &ex) noexcept
- template<Expression Ex>
  auto ceras::transpose (Ex const &ex) noexcept
- auto ceras::img2col (unsigned long const row_kernel, unsigned long col_kernel=-1, unsigned long const row_padding=0, unsigned long col_padding=0, unsigned long const row_stride=1, unsigned long const col↩ _stride=1, unsigned long const row_dilation=1, unsigned long const col_dilation=1) noexcept

- auto ceras::conv2d (unsigned long row_input, unsigned long col_input, unsigned long const row_stride=1, unsigned long const col_stride=1, unsigned long const row_dilation=1, unsigned long const col_dilation=1, std::string const &padding="valid") noexcept
- template<typename T >
  requires std::floating_point< T > auto ceras::drop_out (T const factor) noexcept
- auto ceras::max_pooling_2d (unsigned long stride) noexcept
- auto ceras::average_pooling_2d (unsigned long stride) noexcept
- auto ceras::up_sampling_2d (unsigned long stride) noexcept
- template<typename T = double>
  requires std::floating_point< T > auto ceras::normalization_batch (T const momentum=0.98) noexcept
- template<typename T >
  requires std::floating_point< T > auto ceras::batch_normalization (T const momentum=0.98) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::concatenate (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- auto ceras::concatenate (unsigned long axe=-1)
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::concat (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- auto ceras::concat (unsigned long axe=-1)
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::maximum (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<typename T = float>
  requires std::floating_point< T > auto ceras::random_normal_like (T mean=0.0, T stddev=1.0) noexcept
- template<Expression Ex>
  auto ceras::ones_like (Ex const &ex) noexcept
- template<Expression Lhs_Expression, Expression Rhs_Expression>
  constexpr auto ceras::equal (Lhs_Expression const &lhs_ex, Rhs_Expression const &rhs_ex) noexcept
- template<Expression Ex>
  constexpr auto ceras::sign (Ex const &ex) noexcept
- auto ceras::zero_padding_2d (std::vector< unsigned long > const &padding) noexcept

  Zero-padding layer for 2D input. The input should have 4-dimensions: `(batch_size, row, col, channel)`*. The output has 4-dimensions:* `(batch_size, new_row, new_col, channel)`*.*

Variables

- static constexpr auto ceras::make_unary_operator
- static constexpr auto ceras::make_binary_operator
- template<class T >
  constexpr bool ceras::is_unary_operator_v = is_unary_operator<T>::value
- template<typename T >
  concept ceras::Unary_Operator = is_unary_operator_v<T>

  A type that represents an unary operator.
- template<class T >
  constexpr bool ceras::is_binary_operator_v = is_binary_operator<T>::value
- template<typename T >
  concept ceras::Binary_Operator = is_binary_operator_v<T>

  A type that represents a binary operator.
- template<typename T >
  concept ceras::Operator = Unary_Operator<T> || Binary_Operator<T>

  A type that represents an unary or a binary operator.
- template<typename T >
  concept ceras::Expression = Operator<T> || Variable<T> || Place_Holder<T> || Constant<T> || Value<T>

  A type that represents a unary operator, a binary operator, a variable, a place_holder, a constant or a value.

7.11 /data/structured_↩ folders/workspace/github.repo/ceras/include/optimizer.hpp File Reference

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

Classes

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

Namespaces

- ceras

Typedefs

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

Variables

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

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

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

Classes

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

Namespaces

- ceras

Functions

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

Variables

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

7.13 /data/structured_↩ folders/workspace/github.repo/ceras/include/session.hpp File Reference

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

Classes

- struct ceras::ceras_private::session< Tsor >

Namespaces

- ceras
- ceras::ceras_private

Functions

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

7.14 /data/structured_↩ folders/workspace/github.repo/ceras/include/tensor.hpp File Reference

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

Classes

- struct ceras::tensor< T, Allocator >
- struct ceras::is_tensor< T >
- struct ceras::is_tensor< tensor< T, A > >
- struct ceras::view_2d< T >
- struct ceras::view_3d< T >
- struct ceras::view_4d< T >

Namespaces

- ceras

Typedefs

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

Functions

- template<typename T , typename A = default_allocator<T>>
  constexpr tensor< T, A > ceras::as_tensor (T val) noexcept
- template<Tensor 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>
  requires std::floating_point< typename Tsor::value_type > Tsor::value_type ceras::var (Tsor const &ts) noexcept
- template<Tensor Tsor>
  requires std::floating_point< typename Tsor::value_type > Tsor::value_type ceras::std (Tsor const &ts) noexcept
- template<Tensor Tsor>
  Tsor ceras::max (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept
- template<Tensor Tsor>
  Tsor ceras::min (Tsor const &ts, unsigned long axis, bool keepdims=false) noexcept
- template<typename T , typename A = default_allocator<T>>
  requires std::floating_point< T > tensor< T, A > ceras::linspace (T start, T stop, unsigned long num, bool endpoint=true) noexcept
- template<class _Tp , class _CharT , class _Traits , class _Alloc >
  std::basic_istream< _CharT, _Traits > & ceras::read_tensor (std::basic_istream< _CharT, _Traits > &__is, tensor< _Tp, _Alloc > &__x)
- template<class _Tp , class _CharT , class _Traits , class _Alloc >
  std::basic_ostream< _CharT, _Traits > & ceras::write_tensor (std::basic_ostream< _CharT, _Traits > &__os, tensor< _Tp, _Alloc > const &__x)

- template<typename T , typename A = default_allocator<T>>
  tensor< T, A > ceras::load_tensor (std::string const &file_name)
- template<Tensor Tsor>
  void ceras::save_tensor (std::string const &file_name, Tsor const &tsor)

Variables

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

7.15 /data/structured_↩ folders/workspace/github.repo/ceras/include/value.hpp File Reference

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

Classes

- struct ceras::value< T >
- struct ceras::is_value< T >
- struct ceras::is_value< value< T > >
- struct ceras::tensor_deduction< L, R >

Namespaces

- ceras

Variables

- template<class T >
  constexpr bool ceras::is_value_v = is_value<T>::value
- template<typename T >
  concept ceras::Value = is_value_v<T>

7.16 /data/structured_↩ folders/workspace/github.repo/ceras/include/variable.hpp File Reference

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

Classes

- struct ceras::variable_state< Tsor >
- struct ceras::regularizer< Float >
- struct ceras::variable< Tsor >
- struct ceras::is_variable< T >
- struct ceras::is_variable< variable< Tsor > >

Namespaces

- ceras
- ceras::ceras_private

Functions

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

Variables

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

Index