mini_nn

1.6

Generated by Doxygen 1.8.15

Chapter 1

mini_nn

Author

RuiJian Li, YiFan Cao, YanPeng Hu @email lirj@shanghaitech.edu.cn, caoyf@shanghaitech.edu.cn, huyp@shanghaitech.edu.cn

Version

1.6.0

Date

2019-05-26

Chapter 2

mini_nn

CS133 course project: mini neural network

Generic implementation of a neural network. Build a C++ library that

- can load a pre-trained network definition file
- contains an abstract definition of common layers and the composing elements
  - Linear transformations, convolutions
  - Response functions, output layers
  - Fully connected layers
- initializes concrete layers of the network with a suitable programming technique (e.g. factory method)
- applies it to some data

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

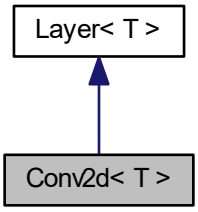
Class Documentation

6.1 Conv2d< T > Class Template Reference

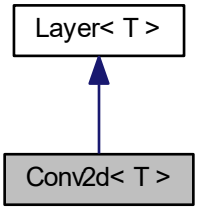
Computes a 2-D convolution given 4-D input and filter tensors.

`#include <convolution.hpp>`

Inheritance diagram for Conv2d< T >:



Collaboration diagram for Conv2d< T >:



Public Types

- typedef xt::xarray< T > Matrix
- typedef Matrix::shape_type Shape

Public Member Functions

- Conv2d ()=default

  Construct a new Conv 2d object.
- virtual ~Conv2d ()=default

  Destroy the Conv 2d object.
- Conv2d (size_t in_channels, size_t out_channels, size_t kernel_size=3, size_t stride=1, size_t padding=0)

  Construct a new Conv 2d< T >:: Conv 2d object.
- virtual Matrix forward (const Matrix &in) override
- virtual Matrix backward (const Matrix &dout) override
- virtual size_t get_fan ()

  Get the fan object.

Protected Attributes

- size_t in_channels_
- size_t out_channels_
- size_t kernel_size_
- size_t padding_
- size_t stride_

6.1.1 Detailed Description

template<typename T>
class Conv2d< T >

Computes a 2-D convolution given 4-D input and filter tensors.

Given an input tensor of shape [batch, in_height, in_width, in_channels] and a filter / kernel tensor of shape [filter_height, filter_width, in_channels, out_channels], this op performs the following:

Flattens the filter to a 2-D matrix with shape [filter_height * filter_width * in_channels, output_channels]. Extracts image patches from the input tensor to form a virtual tensor of shape [batch, out_height, out_width, filter_height * filter_width * in_channels]. For each patch, right-multiplies the filter matrix and the image patch vector.

6.1.2 Member Typedef Documentation

6.1.2.1 Matrix

```
template<typename T >
typedef xt::xarray<T> Conv2d< T >::Matrix
```

6.1.2.2 Shape

```
template<typename T >
typedef Matrix::shape_type Conv2d< T >::Shape
```

6.1.3 Constructor & Destructor Documentation

6.1.3.1 Conv2d() [1/2]

```
template<typename T >
Conv2d< T >::Conv2d ( )  [default]
```

Construct a new Conv 2d object.

6.1.3.2 ~Conv2d()

```
template<typename T >
virtual Conv2d< T >::~Conv2d ( )  [virtual], [default]
```

Destroy the Conv 2d object.

6.1.3.3 Conv2d() [2/2]

```
template<typename T >
Conv2d< T >::Conv2d (
            size_t in_channels,
            size_t out_channels,
            size_t kernel_size = 3,
            size_t stride = 1,
            size_t padding = 0 )
```

Construct a new Conv 2d< T>:: Conv 2d object.

Template Parameters

T	

Parameters

in_channels	: It refers to the input image that needs to be convolved. It is required to be a Tensor with a shape such as [batch, in_height, in_width, in_channels]. The specific meaning is [the number of pictures of a batch during training, the height of the picture, the width of the image, the number of image channels.], note that this is a 4D Tensor,

Parameters

out_channels	
kernel_size	size of the kernel
stride	The convolution step in each dimension of the image, this is a one-dimensional vector, with length 4
padding	This value determines the different convolution methods

6.1.4 Member Function Documentation

6.1.4.1 backward()

```
template<typename T >
xt::xarray< T > Conv2d< T >::backward (
            const Matrix & dout )  [override], [virtual]
```

Template Parameters

T	

Parameters

dout	: the backward input

Returns

xt::xarray<T>

only update if net is already set

Implements Layer< T >.

6.1.4.2 forward()

```
template<typename T >
xt::xarray< T > Conv2d< T >::forward (
            const Matrix & in )  [override], [virtual]
```

Template Parameters

T	

Parameters

in	:input for the network

Returns

xt::xarray<T>

prepare for output

Implements Layer< T >.

6.1.4.3 get_fan()

```
template<typename T >
size_t Conv2d< T >::get_fan ( )  [virtual]
```

Get the fan object.

get the fan of Conv2d

Returns

size_t

Template Parameters

T	

Returns

size_t

Reimplemented from Layer< T >.

6.1.5 Member Data Documentation

6.1.5.1 in_channels_

```
template<typename T >
size_t Conv2d< T >::in_channels_  [protected]
```

6.1.5.2 kernel_size_

```
template<typename T >
size_t Conv2d< T >::kernel_size_  [protected]
```

6.1.5.3 out_channels_

```
template<typename T >
size_t Conv2d< T >::out_channels_  [protected]
```

6.1.5.4 padding_

```
template<typename T >
size_t Conv2d< T >::padding_  [protected]
```

6.1.5.5 stride_

```
template<typename T >
size_t Conv2d< T >::stride_  [protected]
```

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

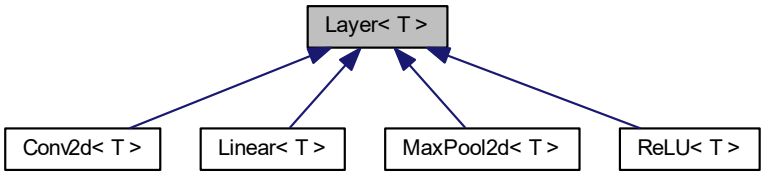
- layer/convolution.hpp
- layer/convolution_impl.hpp

6.2 Layer< T > Class Template Reference

the class of the layer

```
#include <layer_base.hpp>
```

Inheritance diagram for Layer< T >:



Public Types

- typedef xt::xarray< T > Matrix
- typedef Matrix::shape_type Shape

Public Member Functions

- Layer ()=default

  Construct a new Layer object.
- virtual ~Layer ()=default

  Destroy the Layer object.
- virtual Matrix forward (const Matrix &in)=0

  forward function in the network
- virtual Matrix backward (const Matrix &dout)=0

  backward function in the networ
- virtual Shape weight_shape ()
- virtual Shape bias_shape ()
- virtual void set_weight (const Matrix &W)

  Set the weight object.
- virtual void set_bias (const Matrix &b)

  Set the bias object.
- virtual void set_network (Network< T > *net)

  Set the network object.
- virtual LAYER_TYPE get_type ()

  Get the type object.
- virtual size_t get_fan ()

  Get the fan object.

Protected Attributes

- LAYER_TYPE layer_type_
- Matrix in_
- Matrix din_
- Matrix W_
- Matrix dW_
- Matrix b_
- Matrix db_
- Network< T > * net_

6.2.1 Detailed Description

template<typename T>
class Layer< T >

the class of the layer

6.2.2 Member Typedef Documentation

6.2.2.1 Matrix

```
template<typename T>
typedef xt::xarray<T> Layer< T >::Matrix
```

6.2.2.2 Shape

```
template<typename T>
typedef Matrix::shape_type Layer< T >::Shape
```

6.2.3 Constructor & Destructor Documentation

6.2.3.1 Layer()

```
template<typename T>
Layer< T >::Layer ( )  [default]
```

Construct a new Layer object.

6.2.3.2 ~Layer()

```
template<typename T>
virtual Layer< T >::~Layer ( )  [virtual], [default]
```

Destroy the Layer object.

6.2.4 Member Function Documentation

6.2.4.1 backward()

```
template<typename T>
virtual Matrix Layer< T >::backward (
            const Matrix & dout )  [pure virtual]
```

backward function in the networ

Parameters

dout	

Returns

Matrix

Implemented in Conv2d< T >, ReLU< T >, Linear< T >, and MaxPool2d< T >.

6.2.4.2 bias_shape()

```
template<typename T>
virtual Shape Layer< T >::bias_shape ( )  [inline], [virtual]
```

Returns

Shape

6.2.4.3 forward()

```
template<typename T>
virtual Matrix Layer< T >::forward (
            const Matrix & in )  [pure virtual]
```

forward function in the network

Parameters

in	

Returns

Matrix

Implemented in Conv2d< T >, ReLU< T >, Linear< T >, and MaxPool2d< T >.

6.2.4.4 get_fan()

```
template<typename T>
virtual size_t Layer< T >::get_fan ( )  [inline], [virtual]
```

Get the fan object.

Returns

size_t

Reimplemented in Conv2d< T >, and Linear< T >.

6.2.4.5 get_type()

```
template<typename T>
virtual LAYER_TYPE Layer< T >::get_type ( )  [inline], [virtual]
```

Get the type object.

Returns

LAYER_TYPE

6.2.4.6 set_bias()

```
template<typename T>
virtual void Layer< T >::set_bias (
            const Matrix & b )  [inline], [virtual]
```

Set the bias object.

Parameters

b	

6.2.4.7 set_network()

```
template<typename T>
virtual void Layer< T >::set_network (
            Network< T > * net )  [inline], [virtual]
```

Set the network object.

Parameters

net	

6.2.4.8 set_weight()

```
template<typename T>
virtual void Layer< T >::set_weight (
            const Matrix & W )  [inline], [virtual]
```

Set the weight object.

Parameters

W	

6.2.4.9 weight_shape()

```
template<typename T>
virtual Shape Layer< T >::weight_shape ( )  [inline], [virtual]
```

Returns

Shape

6.2.5 Member Data Documentation

6.2.5.1 b_

```
template<typename T>
Matrix Layer< T >::b_  [protected]
```

6.2.5.2 db_

```
template<typename T>
Matrix Layer< T >::db_  [protected]
```

6.2.5.3 din_

```
template<typename T>
Matrix Layer< T >::din_  [protected]
```

6.2.5.4 dW_

```
template<typename T>
Matrix Layer< T >::dW_  [protected]
```

6.2.5.5 in_

```
template<typename T>
Matrix Layer< T >::in_  [protected]
```

6.2.5.6 layer_type_

```
template<typename T>
LAYER_TYPE Layer< T >::layer_type_  [protected]
```

6.2.5.7 net_

```
template<typename T>
Network<T>* Layer< T >::net_  [protected]
```

6.2.5.8 W_

```
template<typename T>
Matrix Layer< T >::W_  [protected]
```

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

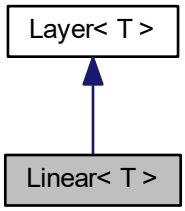
- layer/layer_base.hpp

6.3 Linear< T > Class Template Reference

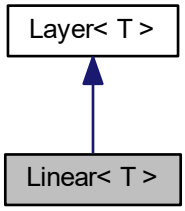
the layer class which inherits the linear class

`#include <linear.hpp>`

Inheritance diagram for Linear< T >:



Collaboration diagram for Linear< T >:



Public Types

- typedef xt::xarray< T > Matrix
- typedef Matrix::shape_type Shape

Public Member Functions

- Linear ()=default

  Construct a new Linear object.
- virtual ~Linear ()=default

  Destroy the Linear object.
- Linear (size_t in_dims, size_t out_dims)

  Construct a new Linear object.

- virtual Matrix forward (const Matrix &in) override

  forward in the network
- virtual Matrix backward (const Matrix &dout) override

  backward in the network
- virtual size_t get_fan ()

  Get the fan object.

Protected Attributes

- Matrix in_reshape_
- size_t in_dims_
- size_t out_dims_

6.3.1 Detailed Description

template<typename T>
class Linear< T >

the layer class which inherits the linear class

Template Parameters

T	

6.3.2 Member Typedef Documentation

6.3.2.1 Matrix

```
template<typename T >
typedef xt::xarray<T> Linear< T >::Matrix
```

6.3.2.2 Shape

```
template<typename T >
typedef Matrix::shape_type Linear< T >::Shape
```

6.3.3 Constructor & Destructor Documentation

6.3.3.1 Linear() [1/2]

```
template<typename T >
Linear< T >::Linear ( )  [default]
```

Construct a new Linear object.

6.3.3.2 ~Linear()

```
template<typename T >
virtual Linear< T >::~Linear ( )  [virtual], [default]
```

Destroy the Linear object.

6.3.3.3 Linear() [2/2]

```
template<typename T >
Linear< T >::Linear (
            size_t in_dims,
            size_t out_dims )
```

Construct a new Linear object.

Construct a new Linear< T>:: Linear object.

Parameters

in_dims	in dimensions
out_dims	out dimensions

Template Parameters

T	

Parameters

in_dims	: in dimensions
out_dims	: out dimensions

6.3.4 Member Function Documentation

6.3.4.1 backward()

```
template<typename T >
xt::xarray< T > Linear< T >::backward (
            const Matrix & dout )  [override], [virtual]
```

backward in the network

the implementation of the backward function

Parameters

dout	

Returns

Matrix

Template Parameters

T	

Parameters

dout	

Returns

xt::xarray<T>

Implements Layer< T >.

6.3.4.2 forward()

```
template<typename T >
xt::xarray< T > Linear< T >::forward (
            const Matrix & in )  [override], [virtual]
```

forward in the network

the implementation of the forward function

Parameters

in	

Returns

Matrix

Template Parameters

T	

Parameters

in	

Returns

xt::xarray<T>

Implements Layer< T >.

6.3.4.3 get_fan()

```
template<typename T >
size_t Linear< T >::get_fan ( )  [virtual]
```

Get the fan object.

get the fan of the network

Returns

size_t

Template Parameters

T	

Returns

size_t

Reimplemented from Layer< T >.

6.3.5 Member Data Documentation

6.3.5.1 in_dims_

```
template<typename T >
size_t Linear< T >::in_dims_  [protected]
```

6.3.5.2 in_reshape_

```
template<typename T >
Matrix Linear< T >::in_reshape_  [protected]
```

6.3.5.3 out_dims_

```
template<typename T >
size_t Linear< T >::out_dims_  [protected]
```

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

- layer/linear.hpp
- layer/linear_impl.hpp

6.4 Loss< T > Class Template Reference

```
#include <loss.hpp>
```

Public Types

- typedef xt::xarray< T > Matrix
- typedef Matrix::shape_type Shape

Public Member Functions

- Loss ()

  Construct a new Loss< T >:: Loss object.
- Loss (LOSS_TYPE loss_type)

  Construct a new Loss< T >:: Loss object.
- virtual ~Loss ()=default
- virtual LOSS_TYPE get_type ()
- virtual const Matrix & get_grad ()
- virtual T CrossEntropyLoss (const Matrix &scores, const Matrix &target)

Protected Attributes

- Matrix scores_
- Matrix dscores_
- LOSS_TYPE loss_type_

6.4.1 Member Typedef Documentation

6.4.1.1 Matrix

```
template<typename T>
typedef xt::xarray<T> Loss< T >::Matrix
```

6.4.1.2 Shape

```
template<typename T>
typedef Matrix::shape_type Loss< T >::Shape
```

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Loss() [1/2]

```
template<typename T >
Loss< T >::Loss ( )
```

Construct a new Loss< T>:: Loss object.

Template Parameters

T	

default use CROSS_ENTROPY

6.4.2.2 Loss() [2/2]

```
template<typename T >
Loss< T >::Loss (
            LOSS_TYPE loss_type )
```

Construct a new Loss< T>:: Loss object.

Template Parameters

T	

Parameters

loss_type	

6.4.2.3 ~Loss()

```
template<typename T>
virtual Loss< T >::~Loss ( )  [virtual], [default]
```

6.4.3 Member Function Documentation

6.4.3.1 CrossEntropyLoss()

```
template<typename T >
T Loss< T >::CrossEntropyLoss (
            const Matrix & scores,
            const Matrix & target )  [virtual]
```

Template Parameters

T	

Parameters

scores	
target	

Returns

T

/// construct index vector (stupied xt::index_view, maybe bug?)

exp_sum.shape(): [N, 1]

exp_sum.shape(): [N, 1]

loss.shape(): [N, 1]

6.4.3.2 get_grad()

```
template<typename T>
virtual const Matrix& Loss< T >::get_grad ( )  [inline], [virtual]
```

6.4.3.3 get_type()

```
template<typename T>
virtual LOSS_TYPE Loss< T >::get_type ( )  [inline], [virtual]
```

6.4.4 Member Data Documentation

6.4.4.1 dscores_

```
template<typename T>
Matrix Loss< T >::dscores_  [protected]
```

6.4.4.2 loss_type_

```
template<typename T>
LOSS_TYPE Loss< T >::loss_type_  [protected]
```

6.4.4.3 scores_

```
template<typename T>
Matrix Loss< T >::scores_  [protected]
```

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

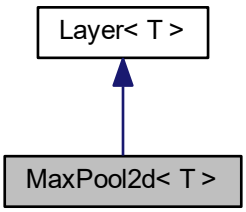
- loss/loss.hpp
- loss/loss_impl.hpp

6.5 MaxPool2d< T > Class Template Reference

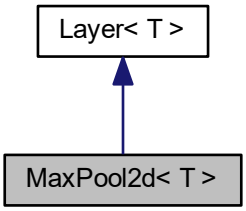
the class for the maxpool

`#include <pooling.hpp>`

Inheritance diagram for MaxPool2d< T >:



Collaboration diagram for MaxPool2d< T >:



Public Types

- typedef xt::xarray< T > Matrix
- typedef Matrix::shape_type Shape

Public Member Functions

- MaxPool2d ()=default

  Construct a new Max Pool 2d object.
- virtual ~MaxPool2d ()=default

  Destroy the Max Pool 2d object.
- MaxPool2d (size_t kernel_size, size_t stride=0, size_t padding=0)

  Construct a new Max Pool 2d object.
- virtual Matrix forward (const Matrix &in) override

  forward function in the network
- virtual Matrix backward (const Matrix &dout) override

  backward function in the network

Protected Attributes

- size_t kernel_size_
- size_t padding_
- size_t stride_

6.5.1 Detailed Description

template<typename T>
class MaxPool2d< T >

the class for the maxpool

Template Parameters

T	

6.5.2 Member Typedef Documentation

6.5.2.1 Matrix

```
template<typename T >
typedef xt::xarray<T> MaxPool2d< T >::Matrix
```

6.5.2.2 Shape

```
template<typename T >
typedef Matrix::shape_type MaxPool2d< T >::Shape
```

6.5.3 Constructor & Destructor Documentation

6.5.3.1 MaxPool2d() [1/2]

```
template<typename T >
MaxPool2d< T >::MaxPool2d ( )  [default]
```

Construct a new Max Pool 2d object.

6.5.3.2 ~MaxPool2d()

```
template<typename T >
virtual MaxPool2d< T >::~MaxPool2d ( )  [virtual], [default]
```

Destroy the Max Pool 2d object.

6.5.3.3 MaxPool2d() [2/2]

```
template<typename T >
MaxPool2d< T >::MaxPool2d (
            size_t kernel_size,
            size_t stride = 0,
            size_t padding = 0 )
```

Construct a new Max Pool 2d object.

Construct a new Max Pool 2d< T>:: Max Pool 2d object.

Parameters

kernel_size	
stride	
padding	

Template Parameters

T	

Parameters

kernel_size	: the size of the kernel
stride	: the convolution is the step size of each dimension of the image, which is a one-dimensional vector.
padding	:This value determines the different convolution methods

6.5.4 Member Function Documentation

6.5.4.1 backward()

```
template<typename T >
xt::xarray< T > MaxPool2d< T >::backward (
            const Matrix & dout )  [override], [virtual]
```

backward function in the network

backford function in the maxpool2d

Parameters

dout	

Returns

Matrix

Template Parameters

T	

Parameters

dout	

Returns

xt::xarray<T>

Implements Layer< T >.

6.5.4.2 forward()

```
template<typename T >
xt::xarray< T > MaxPool2d< T >::forward (
            const Matrix & in )  [override], [virtual]
```

forward function in the network

forward function in the maxpool2d

Parameters

in	

Returns

Matrix

Template Parameters

T	

Parameters

in	: input

Returns

xt::xarray<T>

Implements Layer< T >.

6.5.5 Member Data Documentation

6.5.5.1 kernel_size_

```
template<typename T >
size_t MaxPool2d< T >::kernel_size_  [protected]
```

6.5.5.2 padding_

```
template<typename T >
size_t MaxPool2d< T >::padding_  [protected]
```

6.5.5.3 stride_

```
template<typename T >
size_t MaxPool2d< T >::stride_  [protected]
```

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

- layer/pooling.hpp
- layer/pooling_impl.hpp

6.6 Network< T > Class Template Reference

the class of network

```
#include <layer_base.hpp>
```

Public Types

- typedef xt::xarray< T > Matrix
- typedef Matrix::shape_type Shape

Public Member Functions

- Network ()=default
- virtual ~Network ()=default
- Network< T > & operator<< (Layer< T > *layer)
- Network< T > & operator<< (Layer< T > &layer)
- Network< T > & operator<< (Loss< T > *loss)
- Network< T > & operator<< (Loss< T > &loss)
- virtual Optimizer< T > * get_optimizer ()
- virtual void set_optimizer (Optimizer< T > *opt)
- virtual Matrix predict (const Matrix &in)
- virtual Matrix forward (const Matrix &in, const Matrix &target)
- virtual void backward ()

Protected Attributes

- std::list< Layer< T > * > layers_
- Loss< T > * loss_
- Optimizer< T > * optimizer_

6.6.1 Detailed Description

template<typename T>
class Network< T >

the class of network

6.6.2 Member Typedef Documentation

6.6.2.1 Matrix

```
template<typename T>
typedef xt::xarray<T> Network< T >::Matrix
```

6.6.2.2 Shape

```
template<typename T>
typedef Matrix::shape_type Network< T >::Shape
```

6.6.3 Constructor & Destructor Documentation

6.6.3.1 Network()

```
template<typename T>
Network< T >::Network ( )  [default]
```

6.6.3.2 ~Network()

```
template<typename T>
virtual Network< T >::~Network ( )  [virtual], [default]
```

6.6.4 Member Function Documentation

6.6.4.1 backward()

```
template<typename T >
void Network< T >::backward ( )  [virtual]
```

6.6.4.2 forward()

```
template<typename T >
xt::xarray< T > Network< T >::forward (
            const Matrix & in,
            const Matrix & target )  [virtual]
```

6.6.4.3 get_optimizer()

```
template<typename T>
virtual Optimizer<T>* Network< T >::get_optimizer ( )  [inline], [virtual]
```

6.6.4.4 operator<<() [1/4]

```
template<typename T >
Network< T > & Network< T >::operator<< (
            Loss< T > & loss )
```

6.6.4.5 operator<<() [2/4]

```
template<typename T >
Network< T > & Network< T >::operator<< (
            Layer< T > * layer )
```

6.6.4.6 operator<<() [3/4]

```
template<typename T >
Network< T > & Network< T >::operator<< (
            Layer< T > & layer )
```

6.6.4.7 operator<<() [4/4]

```
template<typename T >
Network< T > & Network< T >::operator<< (
            Loss< T > * loss )
```

6.6.4.8 predict()

```
template<typename T >
xt::xarray< T > Network< T >::predict (
            const Matrix & in )  [virtual]
```

6.6.4.9 set_optimizer()

```
template<typename T>
virtual void Network< T >::set_optimizer (
            Optimizer< T > * opt )  [inline], [virtual]
```

6.6.5 Member Data Documentation

6.6.5.1 layers_

```
template<typename T>
std::list<Layer<T>*> Network< T >::layers_  [protected]
```

6.6.5.2 loss_

```
template<typename T>
Loss<T>* Network< T >::loss_  [protected]
```

6.6.5.3 optimizer_

```
template<typename T>
Optimizer<T>* Network< T >::optimizer_  [protected]
```

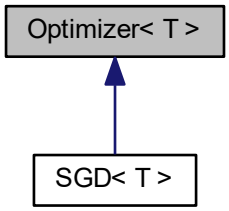
The documentation for this class was generated from the following files:

- layer/layer_base.hpp
- network/network.hpp
- network/network_impl.hpp

6.7 Optimizer< T > Class Template Reference

```
#include <optimizer_base.hpp>
```

Inheritance diagram for Optimizer< T >:



Public Types

- typedef xt::xarray< T > Matrix
- typedef Matrix::shape_type Shape

Public Member Functions

- Optimizer ()=default
- virtual ~Optimizer ()=default
- virtual void update (Matrix &target, const Matrix &grad)=0

Protected Attributes

- T lr_

6.7.1 Member Typedef Documentation

6.7.1.1 Matrix

```
template<typename T>
typedef xt::xarray<T> Optimizer< T >::Matrix
```

6.7.1.2 Shape

```
template<typename T>
typedef Matrix::shape_type Optimizer< T >::Shape
```

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Optimizer()

```
template<typename T>
Optimizer< T >::Optimizer ( )  [default]
```

6.7.2.2 ~Optimizer()

```
template<typename T>
virtual Optimizer< T >::~Optimizer ( )  [virtual], [default]
```

6.7.3 Member Function Documentation

6.7.3.1 update()

```
template<typename T>
virtual void Optimizer< T >::update (
            Matrix & target,
            const Matrix & grad )  [pure virtual]
```

Implemented in SGD< T >.

6.7.4 Member Data Documentation

6.7.4.1 lr_

```
template<typename T>
T Optimizer< T >::lr_  [protected]
```

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

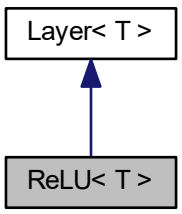
- optimizer/optimizer_base.hpp

6.8 ReLU< T > Class Template Reference

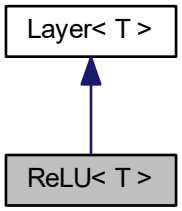
ReLu Class, the rectifier is an activation function.

```
#include <activation.hpp>
```

Inheritance diagram for ReLU< T >:



Collaboration diagram for ReLU< T >:



Public Types

- typedef xt::xarray< T > Matrix
- typedef Matrix::shape_type Shape

Public Member Functions

- ReLU ()

  Construct a new Re L U object.
- virtual ~ReLU ()=default

  Destroy the ReLU object.
- virtual Matrix forward (const Matrix &in) override

  forward function in the network
- virtual Matrix backward (const Matrix &dout) override

  backward function in the network

Additional Inherited Members

6.8.1 Detailed Description

template<typename T>
class ReLU< T >

ReLu Class, the rectifier is an activation function.

Template Parameters

T	

In the context of artificial neural networks, the rectifier is an activation function defined as the positive part of its argument:

{\displaystyle f(x)=x^{+}=\max(0,x)} {\displaystyle f(x)=x^{+}=\max(0,x)}

where x is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification in electrical engineering. This activation function was first introduced to a dynamical network by Hahnloser et al. in 2000 with strong biological motivations and mathematical justifications. It has been demonstrated for the first time in 2011 to enable better training of deeper networks,compared to the widely-used activation functions prior to 2011, e.g., the logistic sigmoid (which is inspired by probability theory; see logistic regression) and its more practical counterpart, the hyperbolic tangent. The rectifier is, as of 2017, the most popular activation function for deep neural networks.

6.8.2 Member Typedef Documentation

6.8.2.1 Matrix

```
template<typename T >
typedef xt::xarray<T> ReLU< T >::Matrix
```

6.8.2.2 Shape

```
template<typename T >
typedef Matrix::shape_type ReLU< T >::Shape
```

6.8.3 Constructor & Destructor Documentation

6.8.3.1 ReLU()

```
template<typename T >
ReLU< T >::ReLU ( )
```

Construct a new Re L U object.

Construct a new Re L U< T>:: Re L U object, the rectifier is an activation function.

Template Parameters

T	

In the context of artificial neural networks, the rectifier is an activation function defined as the positive part of its argument:

{\displaystyle f(x)=x^{+}=\max(0,x)} {\displaystyle f(x)=x^{+}=\max(0,x)}

where x is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification in electrical engineering. This activation function was first introduced to a dynamical network by Hahnloser et al.

in 2000 with strong biological motivations and mathematical justifications. It has been demonstrated for the first time in 2011 to enable better training of deeper networks, compared to the widely-used activation functions prior to 2011, e.g., the logistic sigmoid (which is inspired by probability theory; see logistic regression) and its more practical counterpart, the hyperbolic tangent. The rectifier is, as of 2017, the most popular activation function for deep neural networks.

6.8.3.2 ~ReLU()

```
template<typename T >
virtual ReLU< T >::~ReLU ( )  [virtual], [default]
```

Destroy the ReLU object.

6.8.4 Member Function Documentation

6.8.4.1 backward()

```
template<typename T >
xt::xarray< T > ReLU< T >::backward (
            const Matrix & dout )  [override], [virtual]
```

backward function in the network

Parameters

dout	

Returns

Matrix

Template Parameters

T	

Parameters

dout	

Returns

xt::xarray<T>

Implements Layer< T >.

6.8.4.2 forward()

```
template<typename T >
xt::xarray< T > ReLU< T >::forward (
             const Matrix & in )  [override], [virtual]
```

forward function in the network

forward function

Template Parameters

T	

Parameters

in	

Returns

xt::xarray<T>

Template Parameters

T	

Parameters

in	the input

Returns

xt::xarray<T>

Implements Layer< T >.

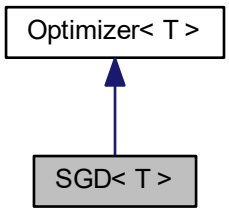
The documentation for this class was generated from the following files:

- layer/activation.hpp
- layer/activation_impl.hpp

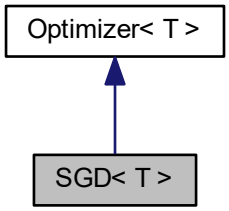
6.9 SGD< T > Class Template Reference

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

Inheritance diagram for SGD< T >:



Collaboration diagram for SGD< T >:



Public Types

- typedef xt::xarray< T > Matrix
- typedef Matrix::shape_type Shape

Public Member Functions

- SGD (T lr=0.1, T momentum=1., T weight_decay=0.)
- virtual ~SGD ()=default
- virtual void update (Matrix &target, const Matrix &grad) override

Protected Attributes

- T momentum_
- T weight_decay_

6.9.1 Member Typedef Documentation

6.9.1.1 Matrix

```
template<typename T >
typedef xt::xarray<T> SGD< T >::Matrix
```

6.9.1.2 Shape

```
template<typename T >
typedef Matrix::shape_type SGD< T >::Shape
```

6.9.2 Constructor & Destructor Documentation

6.9.2.1 SGD()

```
template<typename T >
SGD< T >::SGD (
            T lr = 0.1,
            T momentum = 1.,
            T weight_decay = 0.  )
```

6.9.2.2 ~SGD()

```
template<typename T >
virtual SGD< T >::~SGD ( )  [virtual], [default]
```

6.9.3 Member Function Documentation

6.9.3.1 update()

```
template<typename T >
void SGD< T >::update (
            Matrix & target,
            const Matrix & grad )  [override], [virtual]
```

Implements Optimizer< T >.

6.9.4 Member Data Documentation

6.9.4.1 momentum_

```
template<typename T >
T SGD< T >::momentum_  [protected]
```

6.9.4.2 weight_decay_

```
template<typename T >
T SGD< T >::weight_decay_  [protected]
```

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

- optimizer/optimizer.hpp
- optimizer/optimizer_impl.hpp

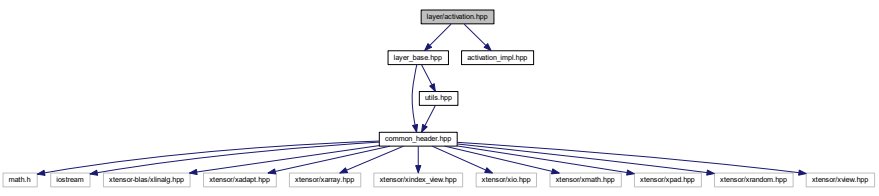
Chapter 7

File Documentation

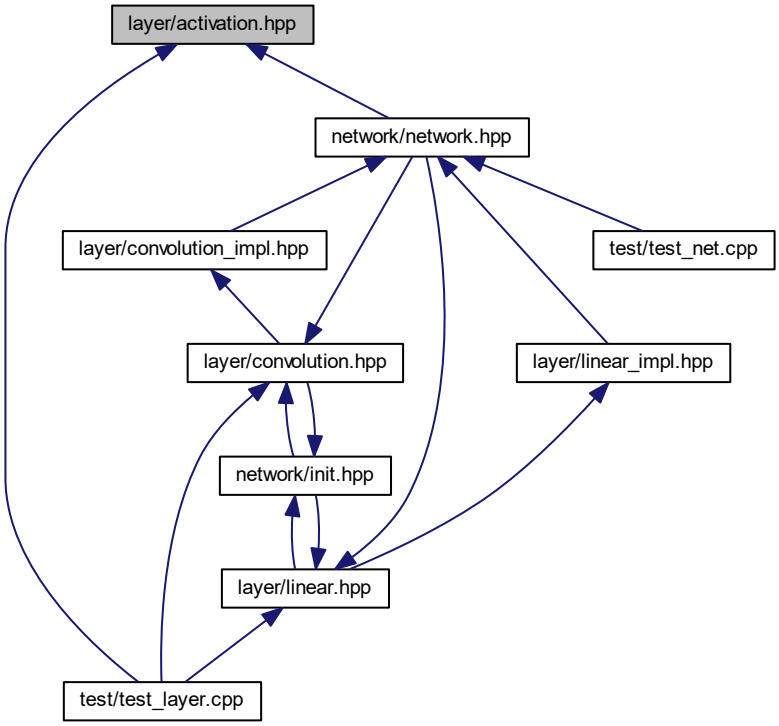
7.1 layer/activation.hpp File Reference

```
#include "layer_base.hpp"
#include "activation_impl.hpp"
```

Include dependency graph for activation.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class ReLU< T >

  ReLu Class, the rectifier is an activation function.

7.1.1 Detailed Description

Author

RuiJian Li(lirj@shanghaitech.edu.cn), YiFan Cao(caoyf@shanghaitech.edu.cn), Yan↩ Peng Hu(huyp@shanghaitech.edu.cn)

Version

1.6.0

Date

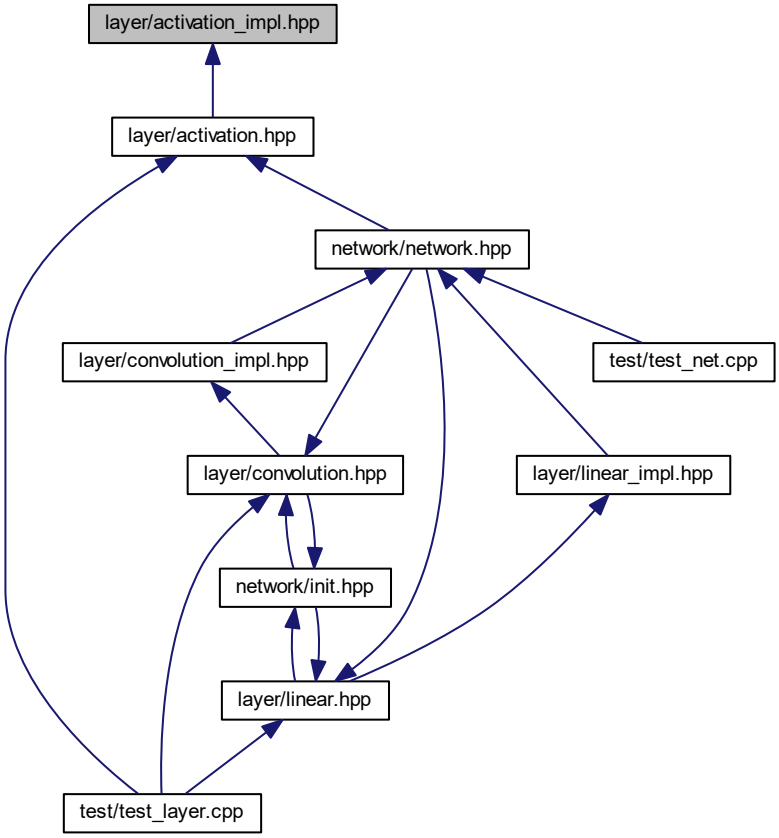
2019-05-30

Copyright

7.2 layer/activation_impl.hpp File Reference

the implentation for the header file of activation function

This graph shows which files directly or indirectly include this file:



7.2.1 Detailed Description

the implentation for the header file of activation function

Author

RuiJian Li, YiFan Cao, YanPeng Hu

Version

1.6.0

Date

2019-05-30

Copyright

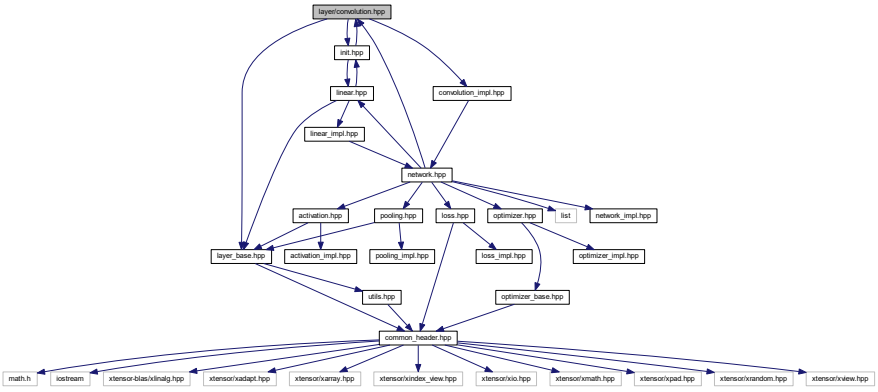
Copyright (c) 2019

7.3 layer/convolution.hpp File Reference

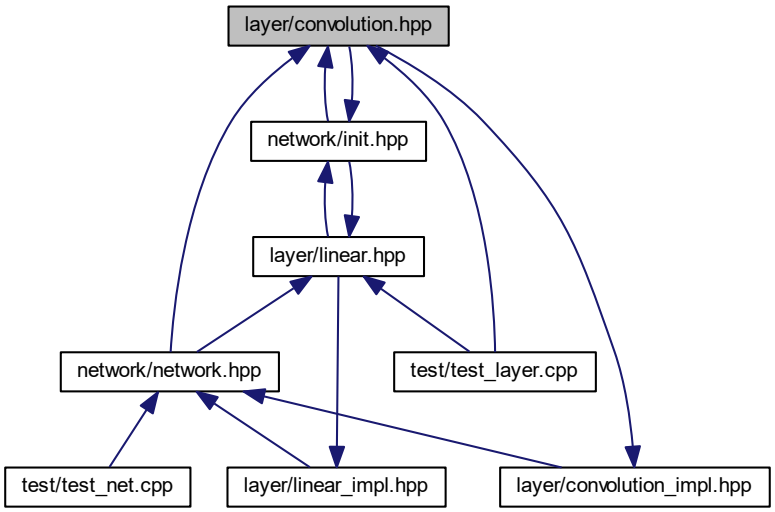
the attribute of the convolution

```
#include "init.hpp"
#include "layer_base.hpp"
#include "convolution_impl.hpp"
```

Include dependency graph for convolution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Conv2d< T >

  Computes a 2-D convolution given 4-D input and filter tensors.

7.3.1 Detailed Description

the attribute of the convolution

Author

RuiJian Li(lirj@shanghaitech.edu.cn), YiFan Cao(caoyf@shanghaitech.edu.cn), Yan↩ Peng Hu(huyp@shanghaitech.edu.cn)

Version

1.6.0

Date

2019-05-30

Copyright

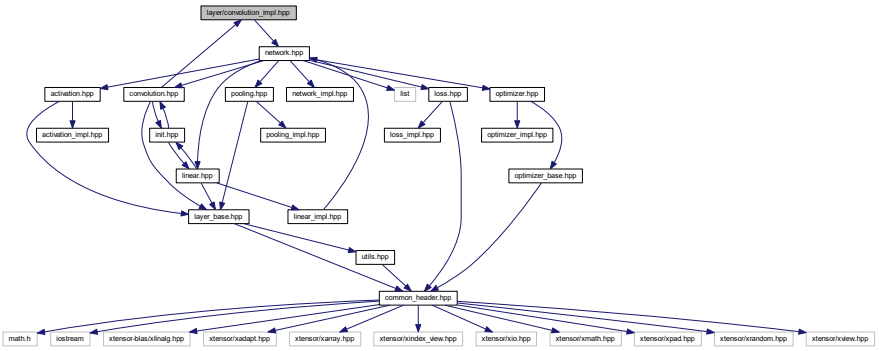
Copyright (c) 2019

7.4 layer/convolution_impl.hpp File Reference

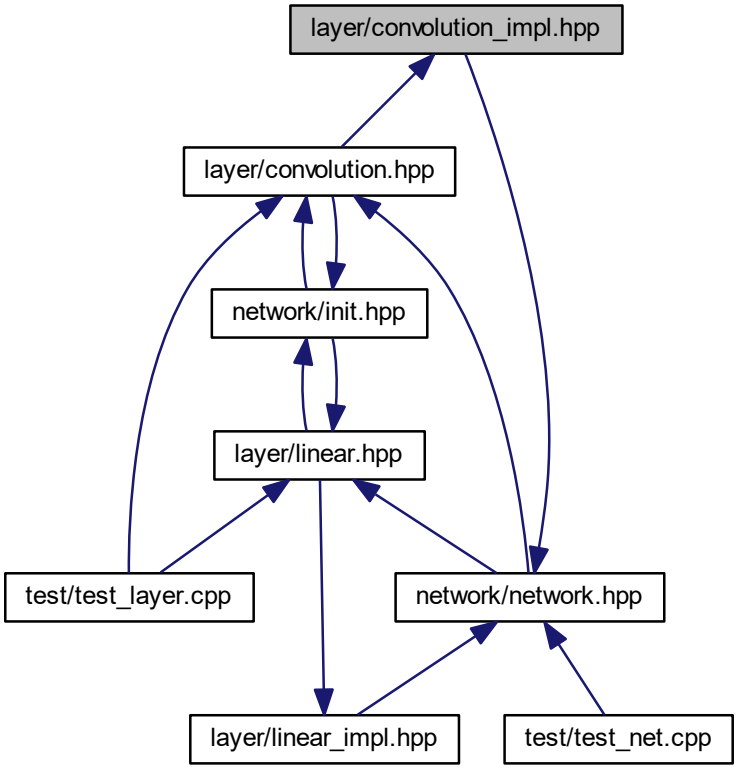
implementation for the convolution

```
#include <network.hpp>
```

Include dependency graph for convolution_impl.hpp:



This graph shows which files directly or indirectly include this file:



7.4.1 Detailed Description

implementation for the convolution

Author

RuiJian Li(lirj@shanghaitech.edu.cn), YiFan Cao(caoyf@shanghaitech.edu.cn), Yan↩ Peng Hu(huyp@shanghaitech.edu.cn)

Version

1.6.0

Date

2019-05-30

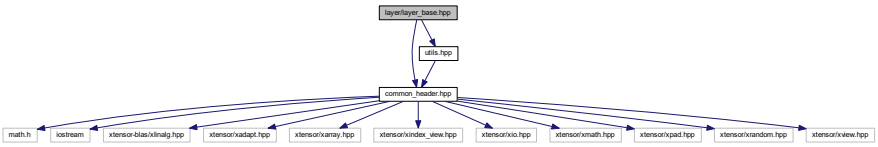
Copyright

7.5 layer/layer_base.hpp File Reference

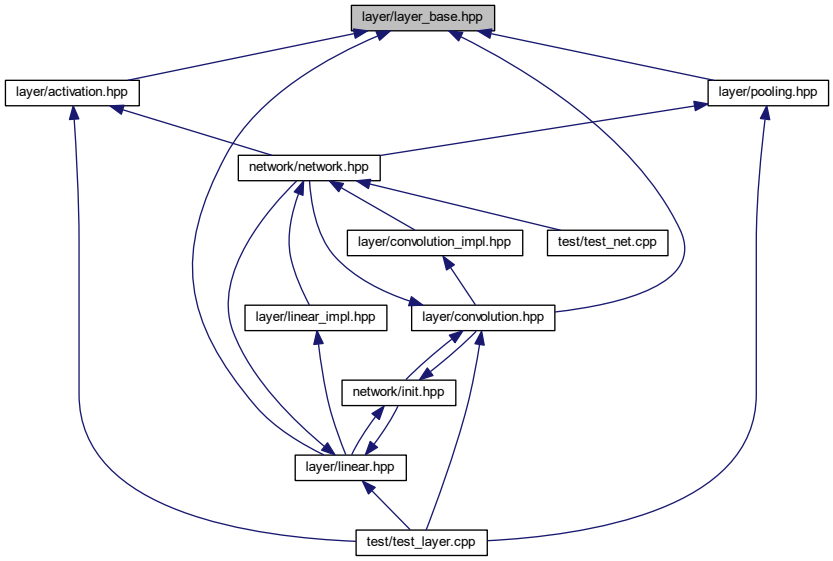
the attribute of the base of the layter

```
#include "common_header.hpp"
#include "utils.hpp"
```

Include dependency graph for layer_base.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Network< T >

  the class of network
- class Layer< T >

  the class of the layer

Enumerations

- enum LAYER_TYPE { CONV, LINEAR, POOL, ACT }

  enumerate the layer type

7.5.1 Detailed Description

the attribute of the base of the layter

Author

RuiJian Li(lirj@shanghaitech.edu.cn), YiFan Cao(caoyf@shanghaitech.edu.cn), Yan↩ Peng Hu(huyp@shanghaitech.edu.cn)

Version

1.6.0

Date

2019-05-30

Copyright

Copyright (c) 2019

7.5.2 Enumeration Type Documentation

7.5.2.1 LAYER_TYPE

```
enum LAYER_TYPE
```

enumerate the layer type

Enumerator

CONV	
LINEAR	
POOL	
ACT	

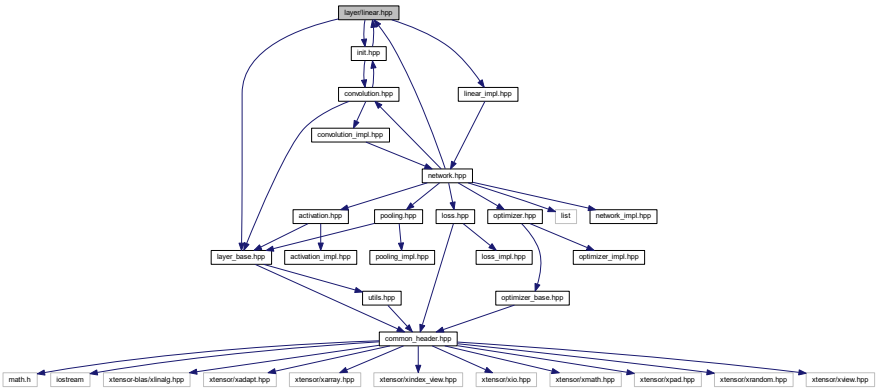
7.6 layer/linear.hpp File Reference

the linear of the header file

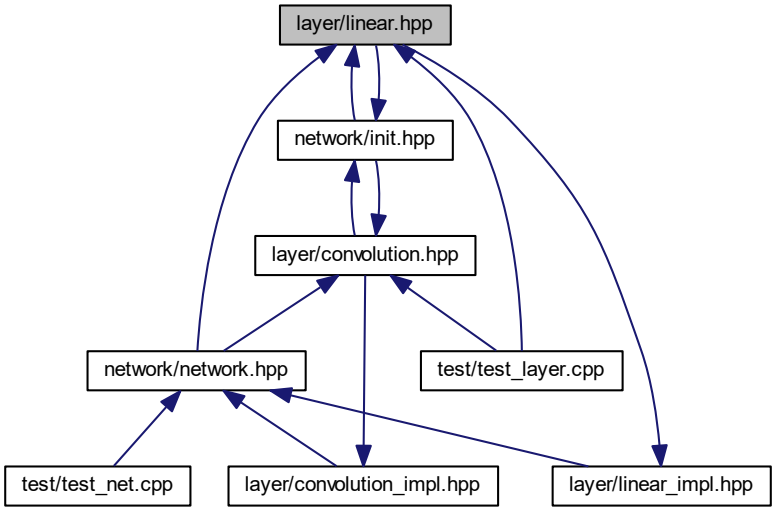
```
#include "init.hpp"
#include "layer_base.hpp"
```

```
#include "linear_impl.hpp"
```

Include dependency graph for linear.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Linear< T >

  the layer class which inherits the linear class

7.6.1 Detailed Description

the linear of the header file

Author

RuiJian Li(lirj@shanghaitech.edu.cn), YiFan Cao(caoyf@shanghaitech.edu.cn), Yan↩ Peng Hu(huyp@shanghaitech.edu.cn)

Version

1.6.0

Date

2019-05-30

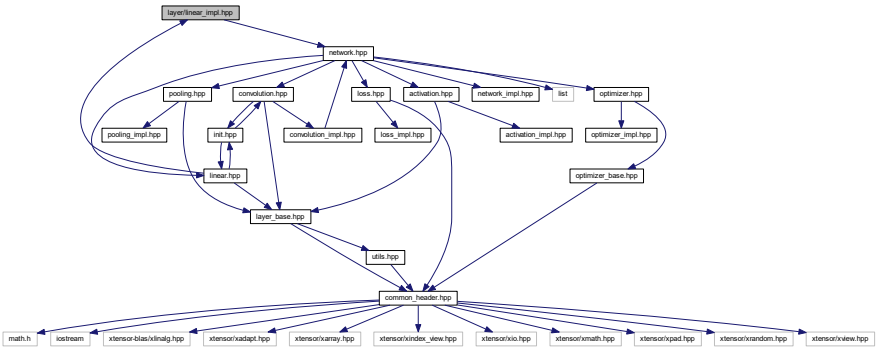
Copyright

7.7 layer/linear_impl.hpp File Reference

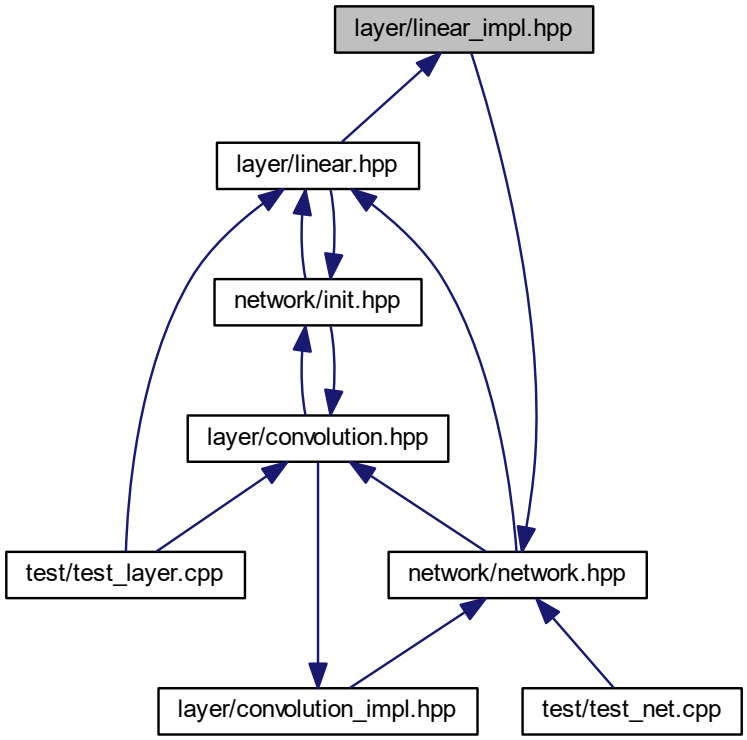
implementation of the linear & forward &backword

```
#include <network.hpp>
```

Include dependency graph for linear_impl.hpp:



This graph shows which files directly or indirectly include this file:



7.7.1 Detailed Description

implementation of the linear & forward &backword

Author

RuiJian Li(lirj@shanghaitech.edu.cn), YiFan Cao(caoyf@shanghaitech.edu.cn), Yan↩ Peng Hu(huyp@shanghaitech.edu.cn)

Version

1.6.0

Date

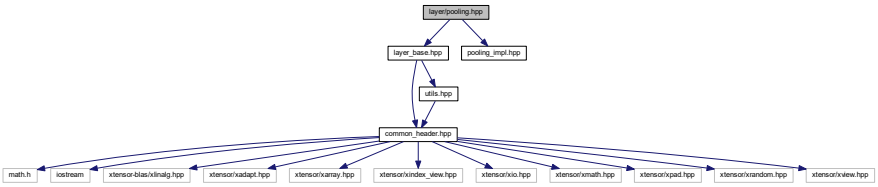
2019-05-30

Copyright

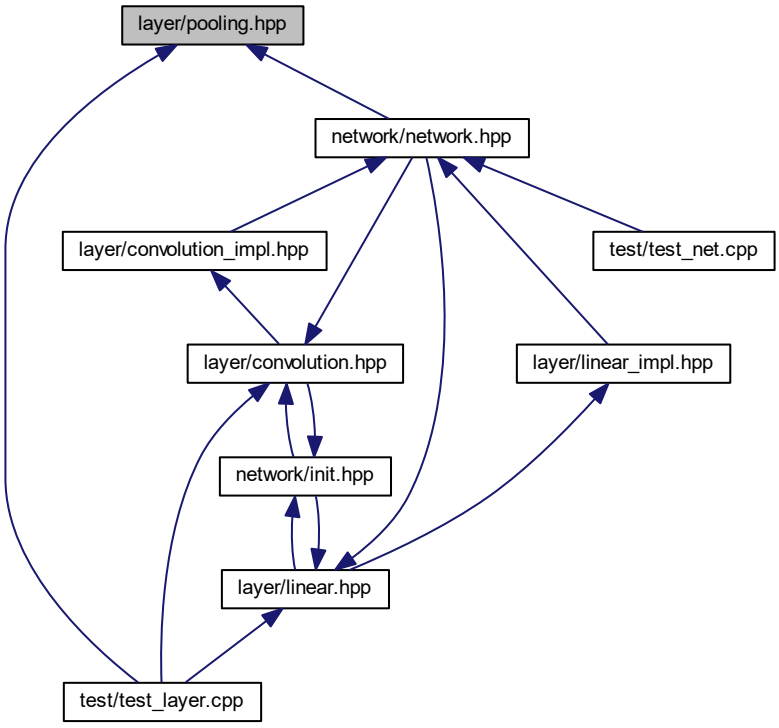
7.8 layer/pooling.hpp File Reference

```
#include "layer_base.hpp"
#include "pooling_impl.hpp"
```

Include dependency graph for pooling.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class MaxPool2d< T >

  the class for the maxpool

7.8.1 Detailed Description

Author

RuiJian Li(lirj@shanghaitech.edu.cn), YiFan Cao(caoyf@shanghaitech.edu.cn), Yan↩ Peng Hu(huyp@shanghaitech.edu.cn)

Version

1.6.0

Date

2019-05-30

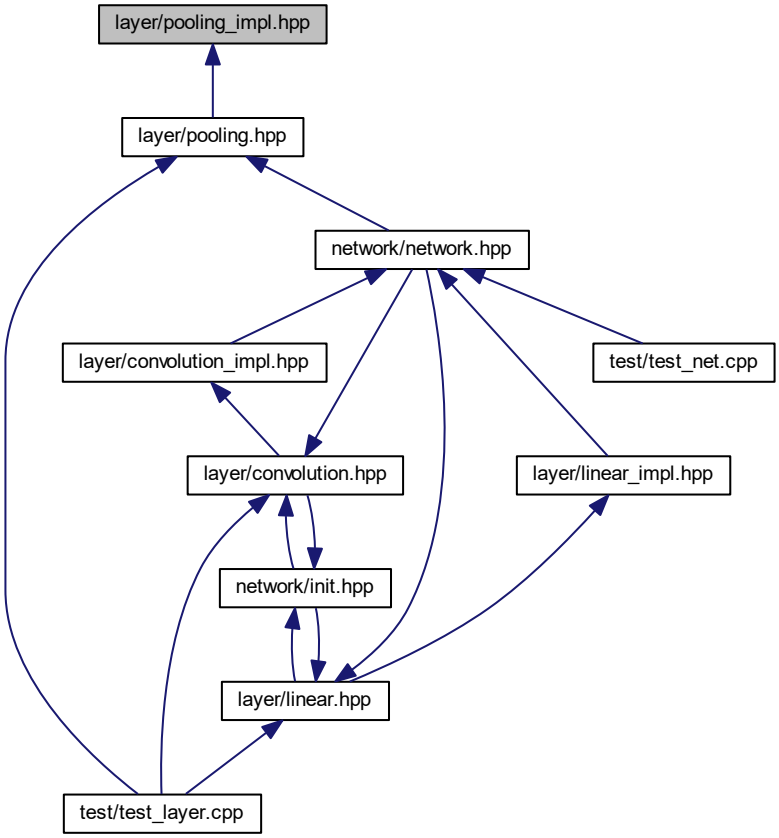
Copyright

Copyright (c) 2019

7.9 layer/pooling_impl.hpp File Reference

the implementation of the pooling

This graph shows which files directly or indirectly include this file:



7.9.1 Detailed Description

the implementation of the pooling

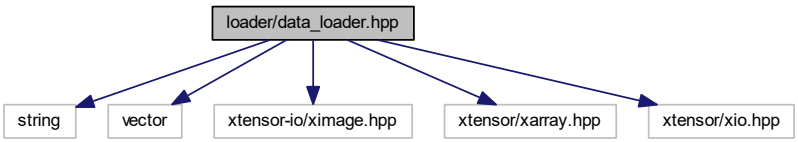
head file

7.10 loader/data_loader.hpp File Reference

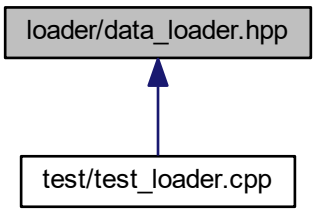
data_loader.hpp

```
#include <string>
#include <vector>
#include "xtensor-io/ximage.hpp"
#include "xtensor/xarray.hpp"
#include "xtensor/xio.hpp"
```

Include dependency graph for data_loader.hpp:



This graph shows which files directly or indirectly include this file:



Functions

- template<typename T >
  void image_normalize (xt::xarray< T > &image, const xt::xarray< T > &mean, const xt::xarray< T > &std_)
- template<typename T >
  xt::xarray< T > load_images (const std::vector< std::string > &paths, const xt::xarray< T > &mean, const xt::xarray< T > &std)

7.10.1 Detailed Description

data_loader.hpp

head file

7.10.2 Function Documentation

7.10.2.1 image_normalize()

```
template<typename T >
void image_normalize (
            xt::xarray< T > & image,
            const xt::xarray< T > & mean,
            const xt::xarray< T > & std_ )
```

7.10.2.2 load_images()

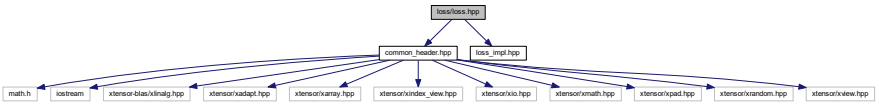
```
template<typename T >
xt::xarray<T> load_images (
            const std::vector< std::string > & paths,
            const xt::xarray< T > & mean,
            const xt::xarray< T > & std )
```

7.11 loader/model_loader.hpp File Reference

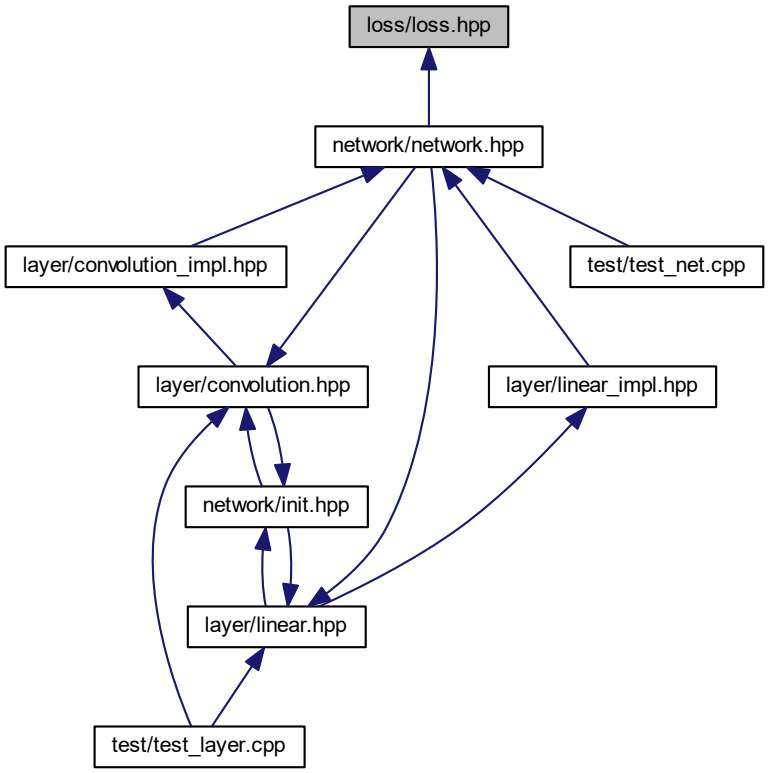
7.12 loss/loss.hpp File Reference

```
#include "common_header.hpp"
#include "loss_impl.hpp"
```

Include dependency graph for loss.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Loss< T >

Enumerations

- enum LOSS_TYPE { CROSS_ENTROPY }

7.12.1 Enumeration Type Documentation

7.12.1.1 LOSS_TYPE

```
enum LOSS_TYPE
```

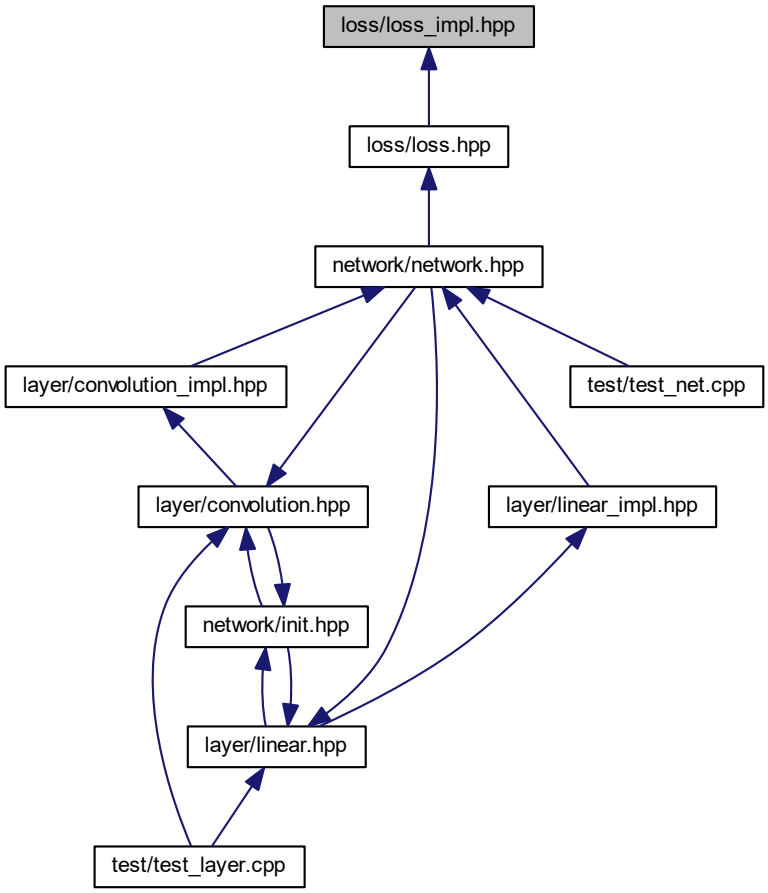
Enumerator

CROSS_ENTROPY	

7.13 loss/loss_impl.hpp File Reference

loss_impl.hpp

This graph shows which files directly or indirectly include this file:



7.13.1 Detailed Description

loss_impl.hpp

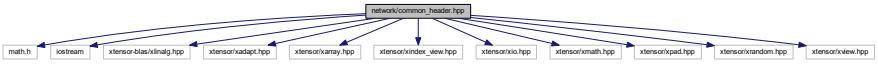
head file

7.14 network/common_header.hpp File Reference

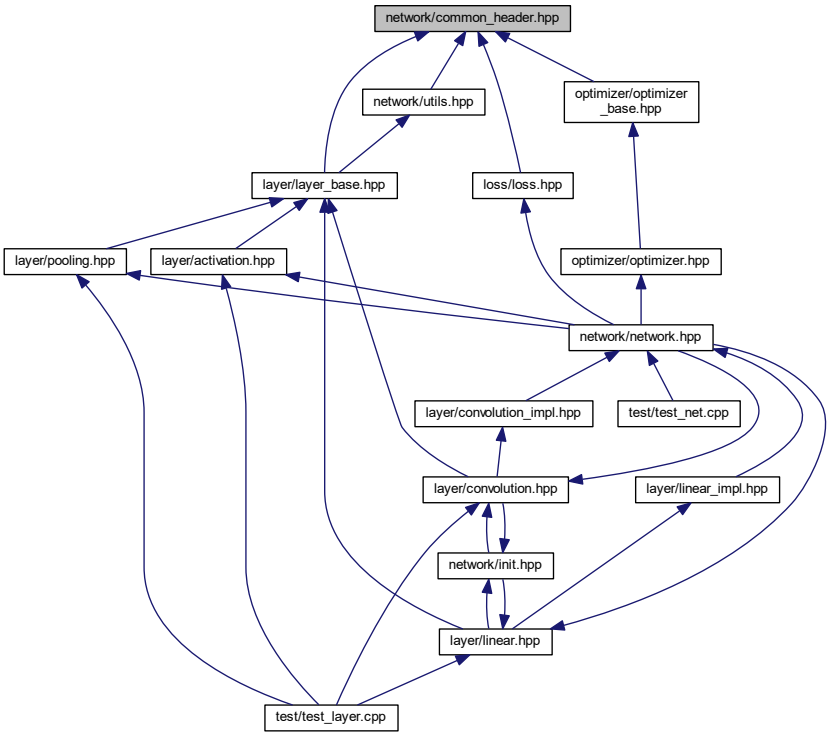
```
#include <math.h>
#include <iostream>
#include "xtensor-blas/xlinalg.hpp"
#include "xtensor/xadapt.hpp"
```

```
#include "xtensor/xarray.hpp"
#include "xtensor/xindex_view.hpp"
#include "xtensor/xio.hpp"
#include "xtensor/xmath.hpp"
#include "xtensor/xpad.hpp"
#include "xtensor/xrandom.hpp"
#include "xtensor/xview.hpp"
```

Include dependency graph for common_header.hpp:



This graph shows which files directly or indirectly include this file:

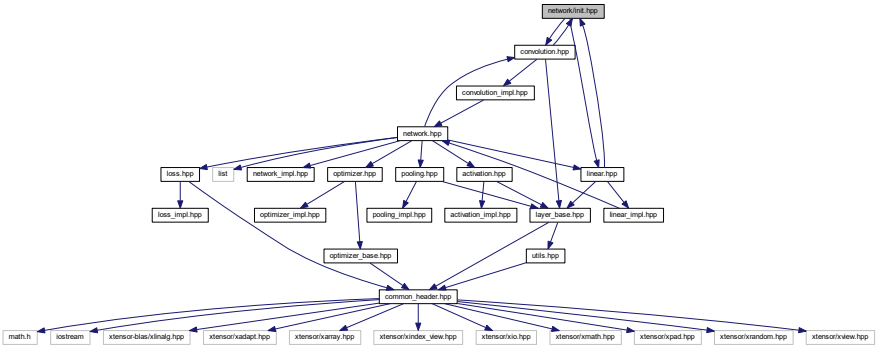


7.15 network/init.hpp File Reference

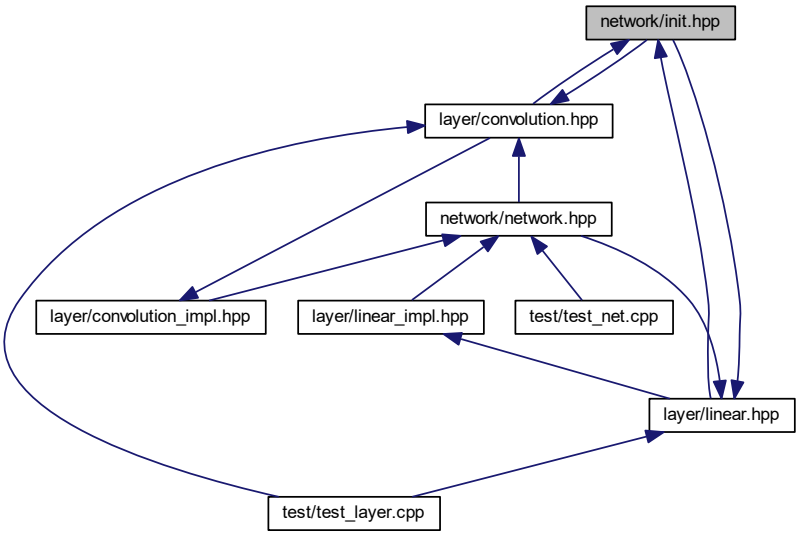
Init the network. Containing two funcitons: kaiming_normal and kaiming_uniform.

```
#include "convolution.hpp"
#include "linear.hpp"
```

Include dependency graph for init.hpp:



This graph shows which files directly or indirectly include this file:



Functions

- template<typename T >
  void kaiming_normal (Layer< T > &layer, std::string mode="ReLU")

  kaiming normal distribution
- template<typename T >
  void kaiming_uniform (Layer< T > &layer, std::string mode="ReLU")

  kaiming uniform distribution

7.15.1 Detailed Description

Init the network. Containing two funcitons: kaiming_normal and kaiming_uniform.

Author

RuiJian Li(lirj@shanghaitech.edu.cn), YiFan Cao(caoyf@shanghaitech.edu.cn), Yan↩ Peng Hu(huyp@shanghaitech.edu.cn)

Version

1.6.0

Date

2019-05-30

Copyright

Copyright (c) 2019

7.15.2 Function Documentation

7.15.2.1 kaiming_normal()

```
template<typename T >
void kaiming_normal (
            Layer< T > & layer,
            std::string mode = "ReLU" )
```

kaiming normal distribution

Template Parameters

T	

Parameters

layer	
mode	

According to the method described by He, K et al. in 'Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification' in 2015, the input tensor or variable is filled with a uniform distribution generation value. The resulting value in the tensor is sampled from U(-bound, bound), where bound = sqrt(2/((1 + a^2) ∗ fan_in)) ∗ sqrt(3). Also known as He initialisation.

7.15.2.2 kaiming_uniform()

```
template<typename T >
void kaiming_uniform (
```

```
Layer< T > & layer,
std::string mode = "ReLU" )
```

kaiming uniform distribution

Template Parameters

T	

Parameters

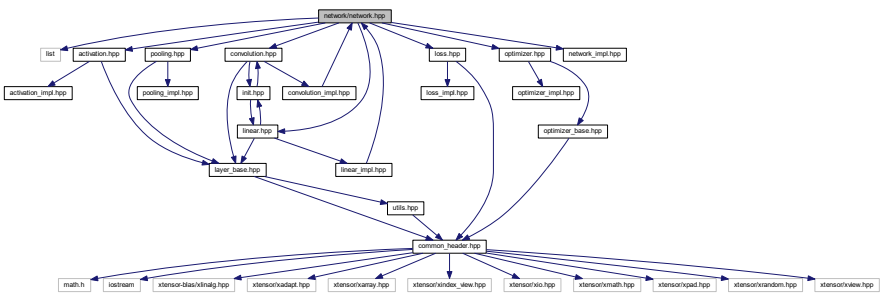
layer	
mode	

According to the method described by He, K et al. in 'Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification' in 2015, the input tensor or variable is filled with a uniform distribution generation value. The resulting value in the tensor is sampled from U(-bound, bound), where bound = sqrt(2/((1 + a^2) $*$ fan_in)) $*$ sqrt(3). Also known as He initialisation.

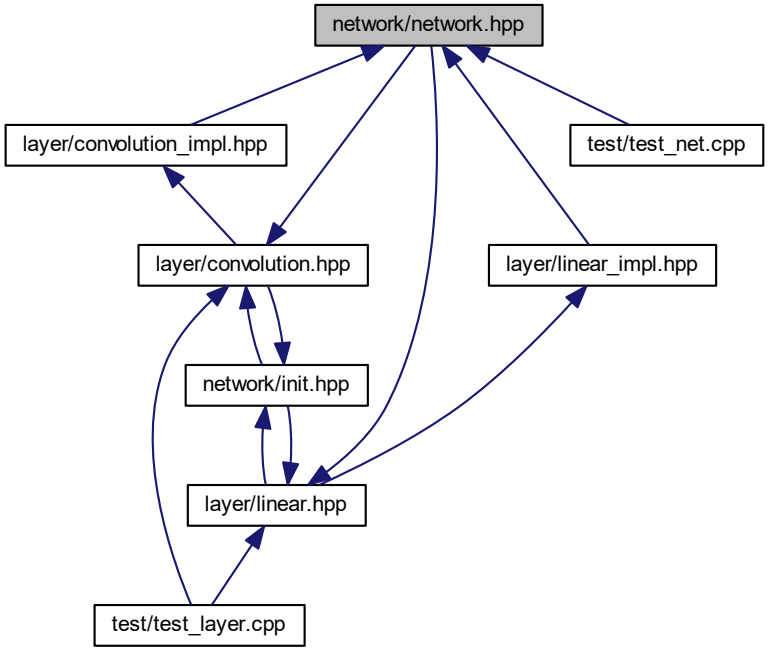
7.16 network/network.hpp File Reference

```
#include <list>
#include "activation.hpp"
#include "convolution.hpp"
#include "linear.hpp"
#include "loss.hpp"
#include "optimizer.hpp"
#include "pooling.hpp"
#include "network_impl.hpp"
```

Include dependency graph for network.hpp:



This graph shows which files directly or indirectly include this file:



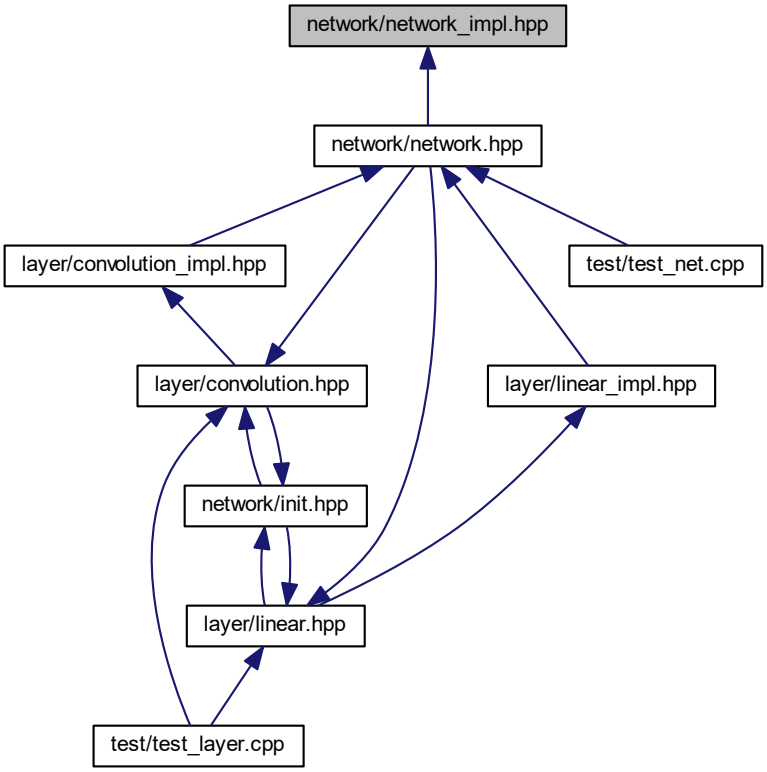
Classes

- class Network< T >

  the class of network

7.17 network/network_impl.hpp File Reference

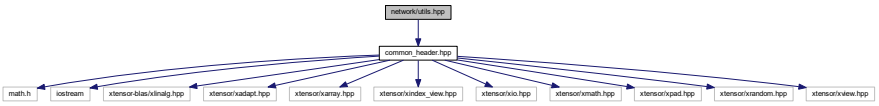
This graph shows which files directly or indirectly include this file:



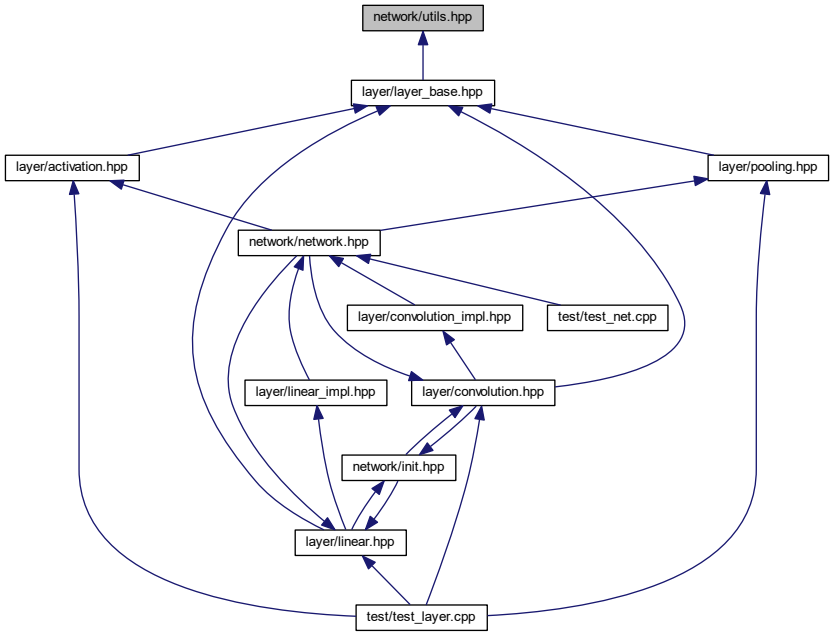
7.18 network/utils.hpp File Reference

```
#include <common_header.hpp>
```

Include dependency graph for utils.hpp:



This graph shows which files directly or indirectly include this file:



Functions

- template<typename T >
  void cout_shape (const xt::xarray< T > &matrix)

7.18.1 Function Documentation

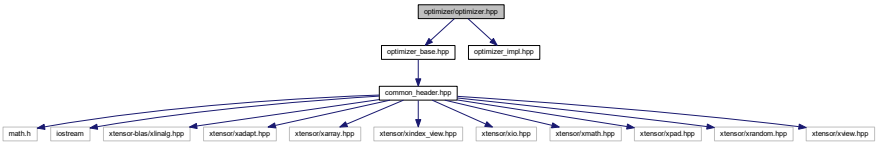
7.18.1.1 cout_shape()

```
template<typename T >
void cout_shape (
            const xt::xarray< T > & matrix )
```

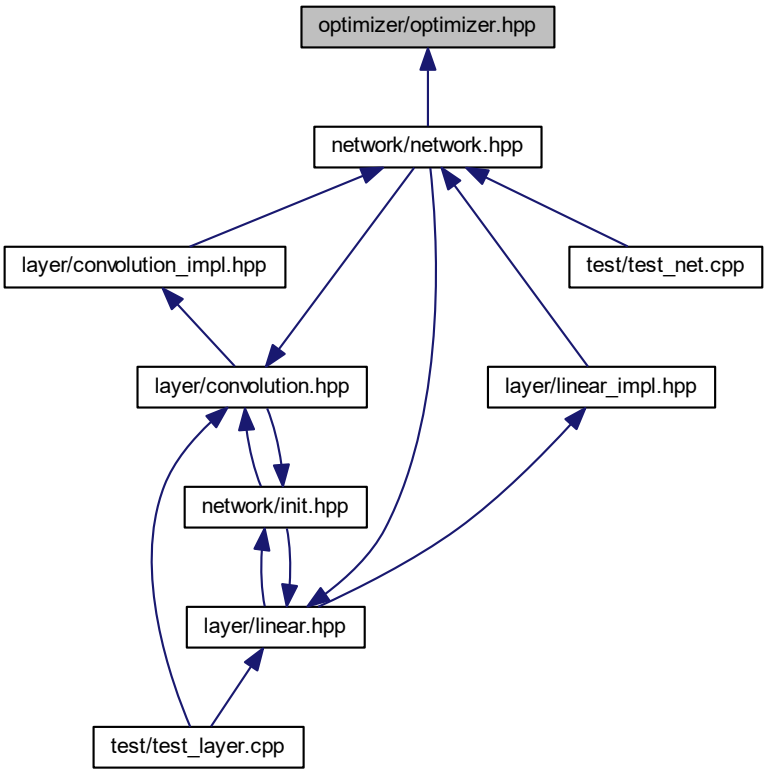
7.19 optimizer/optimizer.hpp File Reference

```
#include "optimizer_base.hpp"
#include "optimizer_impl.hpp"
```

Include dependency graph for optimizer.hpp:



This graph shows which files directly or indirectly include this file:



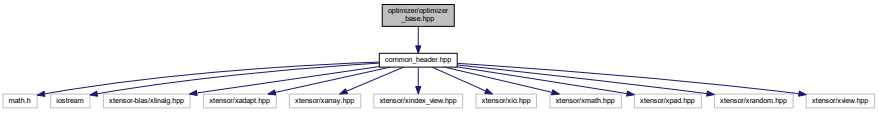
Classes

- class SGD< T >

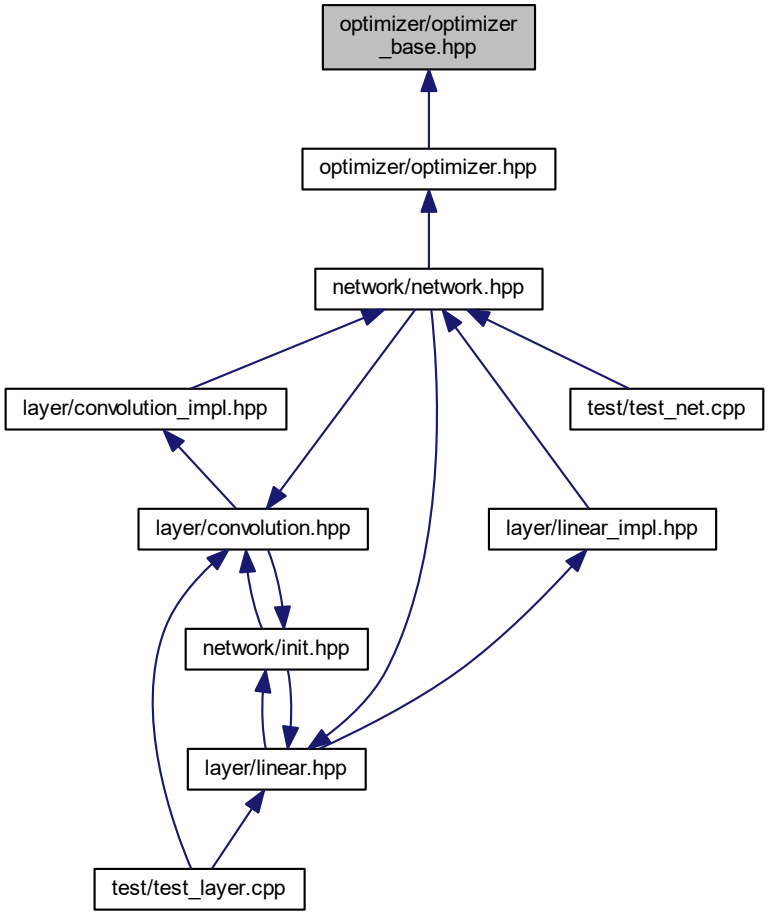
7.20 optimizer/optimizer_base.hpp File Reference

```
#include "common_header.hpp"
```

Include dependency graph for optimizer_base.hpp:



This graph shows which files directly or indirectly include this file:

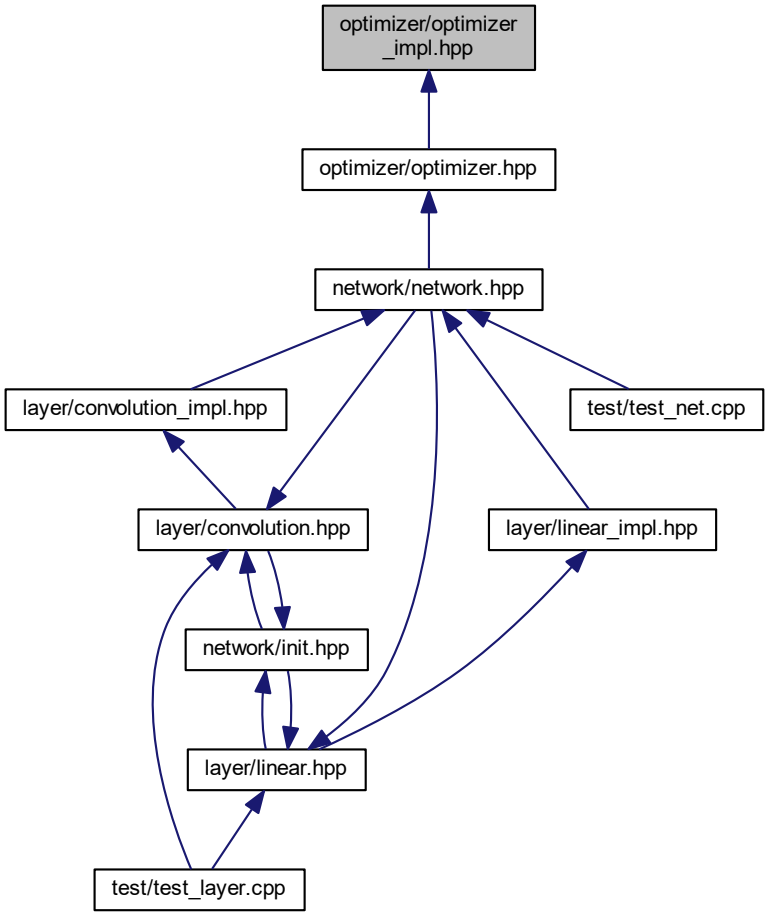


Classes

- class Optimizer< T >

7.21 optimizer/optimizer_impl.hpp File Reference

This graph shows which files directly or indirectly include this file:

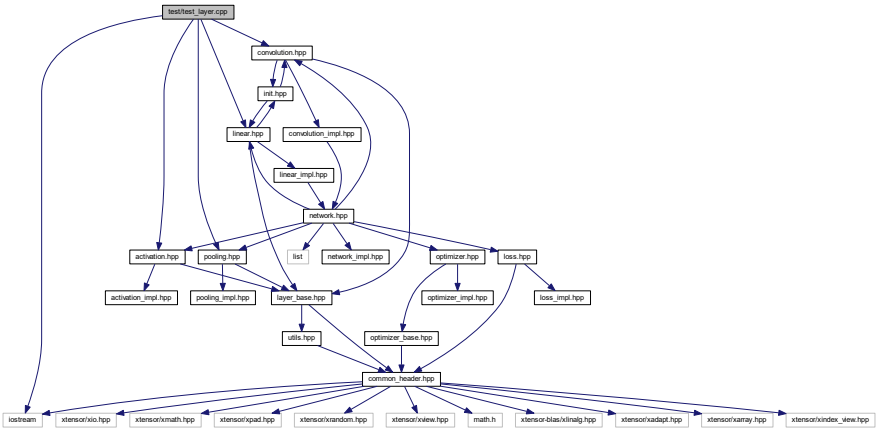


7.22 README.md File Reference

7.23 test/test_layer.cpp File Reference

```
#include "activation.hpp"
#include "convolution.hpp"
#include "linear.hpp"
#include "pooling.hpp"
#include <iostream>
```

Include dependency graph for test_layer.cpp:



Functions

- int main (int argc, char **argv)

7.23.1 Function Documentation

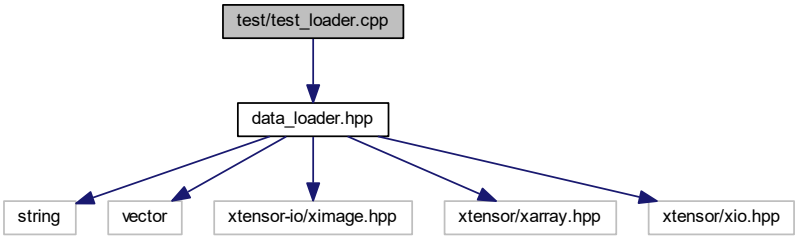
7.23.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

7.24 test/test_loader.cpp File Reference

```
#include "data_loader.hpp"
```

Include dependency graph for test_loader.cpp:



Functions

- int main (int argc, char **argv)

7.24.1 Function Documentation

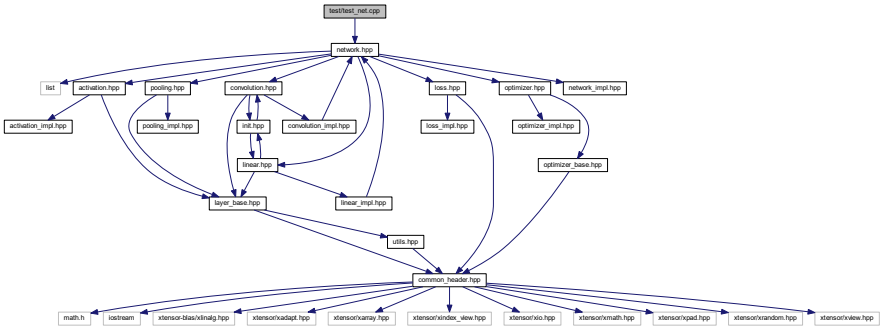
7.24.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

7.25 test/test_net.cpp File Reference

```
#include "network.hpp"
```

Include dependency graph for test_net.cpp:



Functions

- int main (int argc, char **argv)

7.25.1 Function Documentation

7.25.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Index