

mini\_nn

1.6

Generated by Doxygen 1.8.15



<b>1 mini_nn</b>	<b>1</b>
<b>2 mini_nn</b>	<b>3</b>
<b>3 Hierarchical Index</b>	<b>5</b>
3.1 Class Hierarchy	5
<b>4 Class Index</b>	<b>7</b>
4.1 Class List	7
<b>5 File Index</b>	<b>9</b>
5.1 File List	9
<b>6 Class Documentation</b>	<b>11</b>
6.1 Conv2d< T > Class Template Reference	11
6.1.1 Detailed Description	12
6.1.2 Member Typedef Documentation	12
6.1.2.1 Matrix	12
6.1.2.2 Shape	13
6.1.3 Constructor & Destructor Documentation	13
6.1.3.1 Conv2d() [1/2]	13
6.1.3.2 ~Conv2d()	13
6.1.3.3 Conv2d() [2/2]	13
6.1.4 Member Function Documentation	14
6.1.4.1 backward()	14
6.1.4.2 forward()	14
6.1.4.3 get_fan()	15
6.1.5 Member Data Documentation	15
6.1.5.1 in_channels_	15
6.1.5.2 kernel_size_	16
6.1.5.3 out_channels_	16
6.1.5.4 padding_	16
6.1.5.5 stride_	16
6.2 Layer< T > Class Template Reference	16
6.2.1 Detailed Description	17
6.2.2 Member Typedef Documentation	17
6.2.2.1 Matrix	18
6.2.2.2 Shape	18
6.2.3 Constructor & Destructor Documentation	18
6.2.3.1 Layer()	18
6.2.3.2 ~Layer()	18
6.2.4 Member Function Documentation	18
6.2.4.1 backward()	18
6.2.4.2 bias_shape()	19

6.2.4.3 forward()	19
6.2.4.4 get_fan()	19
6.2.4.5 get_type()	20
6.2.4.6 set_bias()	20
6.2.4.7 set_network()	20
6.2.4.8 set_weight()	21
6.2.4.9 weight_shape()	21
6.2.5 Member Data Documentation	21
6.2.5.1 b_	21
6.2.5.2 db_	21
6.2.5.3 din_	21
6.2.5.4 dW_	22
6.2.5.5 in_	22
6.2.5.6 layer_type_	22
6.2.5.7 net_	22
6.2.5.8 W_	22
6.3 Linear< T > Class Template Reference	23
6.3.1 Detailed Description	24
6.3.2 Member Typedef Documentation	24
6.3.2.1 Matrix	24
6.3.2.2 Shape	24
6.3.3 Constructor & Destructor Documentation	24
6.3.3.1 Linear() [1/2]	25
6.3.3.2 ~Linear()	25
6.3.3.3 Linear() [2/2]	25
6.3.4 Member Function Documentation	25
6.3.4.1 backward()	26
6.3.4.2 forward()	26
6.3.4.3 get_fan()	27
6.3.5 Member Data Documentation	27
6.3.5.1 in_dims_	28
6.3.5.2 in_reshape_	28
6.3.5.3 out_dims_	28
6.4 Loss< T > Class Template Reference	28
6.4.1 Member Typedef Documentation	29
6.4.1.1 Matrix	29
6.4.1.2 Shape	29
6.4.2 Constructor & Destructor Documentation	29
6.4.2.1 Loss() [1/2]	29
6.4.2.2 Loss() [2/2]	29
6.4.2.3 ~Loss()	30
6.4.3 Member Function Documentation	30

6.4.3.1 CrossEntropyLoss()	30
6.4.3.2 get_grad()	31
6.4.3.3 get_type()	31
6.4.4 Member Data Documentation	31
6.4.4.1 dscores_	31
6.4.4.2 loss_type_	31
6.4.4.3 scores_	31
6.5 MaxPool2d< T > Class Template Reference	32
6.5.1 Detailed Description	33
6.5.2 Member Typedef Documentation	33
6.5.2.1 Matrix	33
6.5.2.2 Shape	33
6.5.3 Constructor & Destructor Documentation	33
6.5.3.1 MaxPool2d() [1/2]	33
6.5.3.2 ~MaxPool2d()	34
6.5.3.3 MaxPool2d() [2/2]	34
6.5.4 Member Function Documentation	34
6.5.4.1 backward()	34
6.5.4.2 forward()	35
6.5.5 Member Data Documentation	36
6.5.5.1 kernel_size_	36
6.5.5.2 padding_	36
6.5.5.3 stride_	36
6.6 Network< T > Class Template Reference	36
6.6.1 Detailed Description	37
6.6.2 Member Typedef Documentation	37
6.6.2.1 Matrix	37
6.6.2.2 Shape	37
6.6.3 Constructor & Destructor Documentation	37
6.6.3.1 Network()	38
6.6.3.2 ~Network()	38
6.6.4 Member Function Documentation	38
6.6.4.1 backward()	38
6.6.4.2 forward()	38
6.6.4.3 get_optimizer()	38
6.6.4.4 operator<<() [1/4]	38
6.6.4.5 operator<<() [2/4]	39
6.6.4.6 operator<<() [3/4]	39
6.6.4.7 operator<<() [4/4]	39
6.6.4.8 predict()	39
6.6.4.9 set_optimizer()	39
6.6.5 Member Data Documentation	39

6.6.5.1 layers_ . . . . .	39
6.6.5.2 loss_ . . . . .	40
6.6.5.3 optimizer_ . . . . .	40
6.7 Optimizer< T > Class Template Reference . . . . .	40
6.7.1 Member Typedef Documentation . . . . .	41
6.7.1.1 Matrix . . . . .	41
6.7.1.2 Shape . . . . .	41
6.7.2 Constructor & Destructor Documentation . . . . .	41
6.7.2.1 Optimizer() . . . . .	41
6.7.2.2 ~Optimizer() . . . . .	41
6.7.3 Member Function Documentation . . . . .	41
6.7.3.1 update() . . . . .	42
6.7.4 Member Data Documentation . . . . .	42
6.7.4.1 lr_ . . . . .	42
6.8 ReLU< T > Class Template Reference . . . . .	42
6.8.1 Detailed Description . . . . .	43
6.8.2 Member Typedef Documentation . . . . .	44
6.8.2.1 Matrix . . . . .	44
6.8.2.2 Shape . . . . .	44
6.8.3 Constructor & Destructor Documentation . . . . .	44
6.8.3.1 ReLU() . . . . .	44
6.8.3.2 ~ReLU() . . . . .	45
6.8.4 Member Function Documentation . . . . .	45
6.8.4.1 backward() . . . . .	45
6.8.4.2 forward() . . . . .	46
6.9 SGD< T > Class Template Reference . . . . .	46
6.9.1 Member Typedef Documentation . . . . .	47
6.9.1.1 Matrix . . . . .	48
6.9.1.2 Shape . . . . .	48
6.9.2 Constructor & Destructor Documentation . . . . .	48
6.9.2.1 SGD() . . . . .	48
6.9.2.2 ~SGD() . . . . .	48
6.9.3 Member Function Documentation . . . . .	48
6.9.3.1 update() . . . . .	48
6.9.4 Member Data Documentation . . . . .	49
6.9.4.1 momentum_ . . . . .	49
6.9.4.2 weight_decay_ . . . . .	49
<b>7 File Documentation . . . . .</b>	<b>51</b>
7.1 layer/activation.hpp File Reference . . . . .	51
7.1.1 Detailed Description . . . . .	52
7.2 layer/activation_impl.hpp File Reference . . . . .	53

7.2.1 Detailed Description . . . . .	53
7.3 layer/convolution.hpp File Reference . . . . .	54
7.3.1 Detailed Description . . . . .	55
7.4 layer/convolution_impl.hpp File Reference . . . . .	55
7.4.1 Detailed Description . . . . .	56
7.5 layer/layer_base.hpp File Reference . . . . .	57
7.5.1 Detailed Description . . . . .	58
7.5.2 Enumeration Type Documentation . . . . .	58
7.5.2.1 LAYER_TYPE . . . . .	58
7.6 layer/linear.hpp File Reference . . . . .	58
7.6.1 Detailed Description . . . . .	59
7.7 layer/linear_impl.hpp File Reference . . . . .	60
7.7.1 Detailed Description . . . . .	61
7.8 layer/pooling.hpp File Reference . . . . .	62
7.8.1 Detailed Description . . . . .	63
7.9 layer/pooling_impl.hpp File Reference . . . . .	63
7.9.1 Detailed Description . . . . .	64
7.10 loader/data_loader.hpp File Reference . . . . .	64
7.10.1 Detailed Description . . . . .	65
7.10.2 Function Documentation . . . . .	65
7.10.2.1 image_normalize() . . . . .	65
7.10.2.2 load_images() . . . . .	65
7.11 loader/model_loader.hpp File Reference . . . . .	65
7.12 loss/loss.hpp File Reference . . . . .	65
7.12.1 Enumeration Type Documentation . . . . .	66
7.12.1.1 LOSS_TYPE . . . . .	66
7.13 loss/loss_impl.hpp File Reference . . . . .	67
7.13.1 Detailed Description . . . . .	67
7.14 network/common_header.hpp File Reference . . . . .	67
7.15 network/init.hpp File Reference . . . . .	68
7.15.1 Detailed Description . . . . .	69
7.15.2 Function Documentation . . . . .	70
7.15.2.1 kaiming_normal() . . . . .	70
7.15.2.2 kaiming_uniform() . . . . .	70
7.16 network/network.hpp File Reference . . . . .	71
7.17 network/network_impl.hpp File Reference . . . . .	73
7.18 network/utils.hpp File Reference . . . . .	73
7.18.1 Function Documentation . . . . .	74
7.18.1.1 cout_shape() . . . . .	74
7.19 optimizer/optimizer.hpp File Reference . . . . .	74
7.20 optimizer/optimizer_base.hpp File Reference . . . . .	75
7.21 optimizer/optimizer_impl.hpp File Reference . . . . .	77

7.22 README.md File Reference . . . . .	77
7.23 test/test_layer.cpp File Reference . . . . .	77
7.23.1 Function Documentation . . . . .	78
7.23.1.1 main() . . . . .	78
7.24 test/test_loader.cpp File Reference . . . . .	78
7.24.1 Function Documentation . . . . .	79
7.24.1.1 main() . . . . .	79
7.25 test/test_net.cpp File Reference . . . . .	79
7.25.1 Function Documentation . . . . .	79
7.25.1.1 main() . . . . .	79
<b>Index</b>	<b>81</b>



# Chapter 1

## mini\_nn

### Author

RuiJian Li, YiFan Cao, YanPeng Hu @email [lirj@shanghaitech.edu.cn](mailto:lirj@shanghaitech.edu.cn), [caoyf@shanghaitech.edu.cn](mailto:caoyf@shanghaitech.edu.cn), [huyyp@shanghaitech.edu.cn](mailto:huyyp@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:

Layer< T > . . . . .	16
Conv2d< T > . . . . .	11
Linear< T > . . . . .	23
MaxPool2d< T > . . . . .	32
ReLU< T > . . . . .	42
Loss< T > . . . . .	28
Network< T > . . . . .	36
Optimizer< T > . . . . .	40
SGD< T > . . . . .	46



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">Conv2d&lt; T &gt;</a>	Computes a 2-D convolution given 4-D input and filter tensors . . . . .	11
<a href="#">Layer&lt; T &gt;</a>	Class of the layer . . . . .	16
<a href="#">Linear&lt; T &gt;</a>	<a href="#">Layer</a> class which inherits the linear class . . . . .	23
<a href="#">Loss&lt; T &gt;</a>	. . . . .	28
<a href="#">MaxPool2d&lt; T &gt;</a>	Class for the maxpool . . . . .	32
<a href="#">Network&lt; T &gt;</a>	Class of network . . . . .	36
<a href="#">Optimizer&lt; T &gt;</a>	. . . . .	40
<a href="#">ReLU&lt; T &gt;</a>	ReLu Class, the rectifier is an activation function . . . . .	42
<a href="#">SGD&lt; T &gt;</a>	. . . . .	46





## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

layer/activation.hpp	51
layer/activation_impl.hpp	
Implentation for the header file of activation function	53
layer/convolution.hpp	
Attribute of the convolution	54
layer/convolution_impl.hpp	
Implementation for the convolution	55
layer/layer_base.hpp	
Attribute of the base of the layter	57
layer/linear.hpp	
Linear of the header file	58
layer/linear_impl.hpp	
Implementation of the linear & forward &backward	60
layer/pooling.hpp	62
layer/pooling_impl.hpp	
Implementation of the pooling	63
loader/data_loader.hpp	
Data_loader.hpp	64
loader/model_loader.hpp	65
loss/loss.hpp	65
loss/loss_impl.hpp	
Loss_impl.hpp	67
network/common_header.hpp	67
network/init.hpp	
Init the network. Containing two funcitons: kaiming_normal and kaiming_uniform	68
network/network.hpp	71
network/network_impl.hpp	73
network/utils.hpp	73
optimizer/optimizer.hpp	74
optimizer/optimizer_base.hpp	75
optimizer/optimizer_impl.hpp	77
test/test_layer.cpp	77
test/test_loader.cpp	78
test/test_net.cpp	79



## Chapter 6

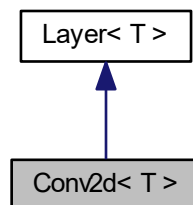
# Class Documentation

### 6.1 Conv2d< T > Class Template Reference

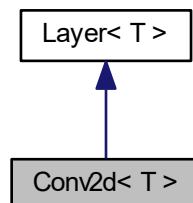
Computes a 2-D convolution given 4-D input and filter tensors.

```
#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

<i>T</i>	
----------	--

#### Parameters

<i>in_channels</i>	: 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

<i>out_channels</i>	
<i>kernel_size</i>	size of the kernel
<i>stride</i>	The convolution step in each dimension of the image, this is a one-dimensional vector, with length 4
<i>padding</i>	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

<i>T</i>	
----------	--

## Parameters

<i>dout</i>	: 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

<i>T</i>	
----------	--

## Parameters

<i>in</i>	: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

<i>T</i>	
----------	--

## 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:

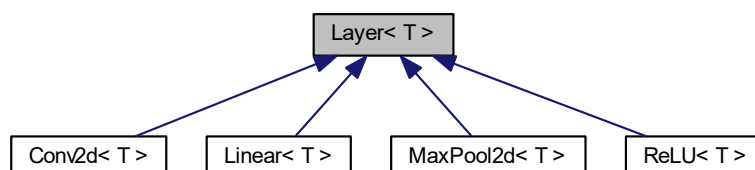
- [layer/convolution.hpp](#)
- [layer/convolution\\_impl.hpp](#)

## 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 network*
- 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**

<i>dout</i>	
-------------	--

**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**

<i>in</i>	
-----------	--

**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**

<i>b</i>	
----------	--

**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**

<i>net</i>	
------------	--

## 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

<i>W</i>	
----------	--

## 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:

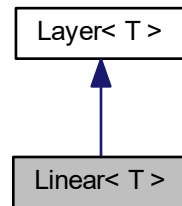
- [layer/layer\\_base.hpp](#)

## 6.3 Linear< T > Class Template Reference

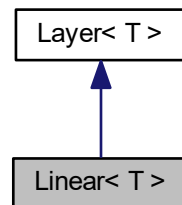
the layer class which inherits the linear class

```
#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

<i>T</i>	
----------	--

### 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**

<i>in_dims</i>	in dimensions
<i>out_dims</i>	out dimensions

**Template Parameters**

<i>T</i>	
----------	--

**Parameters**

<i>in_dims</i>	: in dimensions
<i>out_dims</i>	: 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

<i>dout</i>	
-------------	--

##### Returns

Matrix

##### Template Parameters

<i>T</i>	
----------	--

##### Parameters

<i>dout</i>	
-------------	--

##### 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

<i>in</i>	
-----------	--

**Returns**

Matrix

**Template Parameters**

<i>T</i>	
----------	--

**Parameters**

<i>in</i>	
-----------	--

**Returns**

xt::xarray&lt;T&gt;

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**

<i>T</i>	
----------	--

**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

<i>T</i>	
----------	--

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**

<i>T</i>	
----------	--

**Parameters**

<i>loss_type</i>	
------------------	--

**6.4.2.3  $\sim$ 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**

<i>T</i>	
----------	--

**Parameters**

<i>scores</i>	
<i>target</i>	

**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:

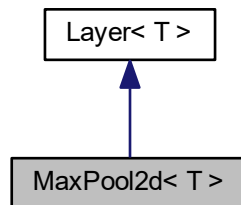
- [loss/loss.hpp](#)
- [loss/loss\\_impl.hpp](#)

## 6.5 MaxPool2d< T > Class Template Reference

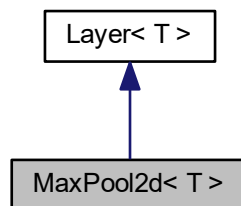
the class for the maxpool

```
#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

<i>T</i>	
----------	--

### 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

<i>kernel_size</i>	
<i>stride</i>	
<i>padding</i>	

#### Template Parameters

<i>T</i>	
----------	--

#### Parameters

<i>kernel_size</i>	: the size of the kernel
<i>stride</i>	: the convolution is the step size of each dimension of the image, which is a one-dimensional vector.
<i>padding</i>	: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

<i>dout</i>	
-------------	--

## Returns

Matrix

## Template Parameters

<i>T</i>	
----------	--

## Parameters

<i>dout</i>	
-------------	--

## Returns

xt::xarray&lt;T&gt;

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

<i>in</i>	
-----------	--

## Returns

Matrix

## Template Parameters

<i>T</i>	
----------	--

## Parameters

<i>in</i>	: 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]
```

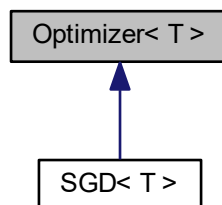
The documentation for this class was generated from the following files:

- [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:

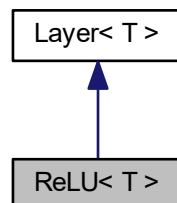
- [optimizer/optimizer\\_base.hpp](#)

## 6.8 ReLU< T > Class Template Reference

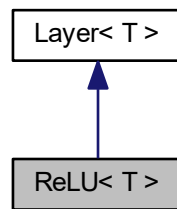
ReLU Class, the rectifier is an activation function.

```
#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

<i>T</i>	
----------	--

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

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

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:

$$f(x) = x^+ = \max(0, x)$$

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

<i>dout</i>	
-------------	--

#### Returns

Matrix

#### Template Parameters

<i>T</i>	
----------	--

#### Parameters

<i>dout</i>	
-------------	--

#### 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

<i>T</i>	
----------	--

##### Parameters

<i>in</i>	
-----------	--

##### Returns

xt::xarray<T>

##### Template Parameters

<i>T</i>	
----------	--

##### Parameters

<i>in</i>	the input
-----------	-----------

##### Returns

xt::xarray<T>

Implements [Layer< T >](#).

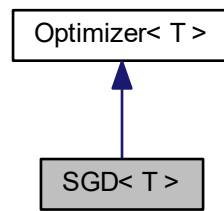
The documentation for this class was generated from the following files:

- [layer/activation.hpp](#)
- [layer/activation\\_impl.hpp](#)

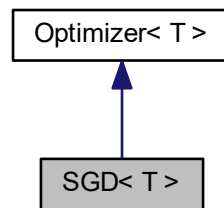
## 6.9 SGD< T > Class Template Reference

```
#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](#)

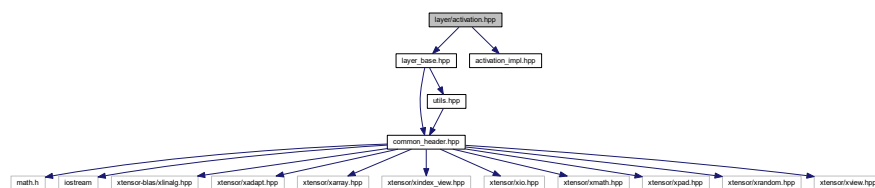


## Chapter 7

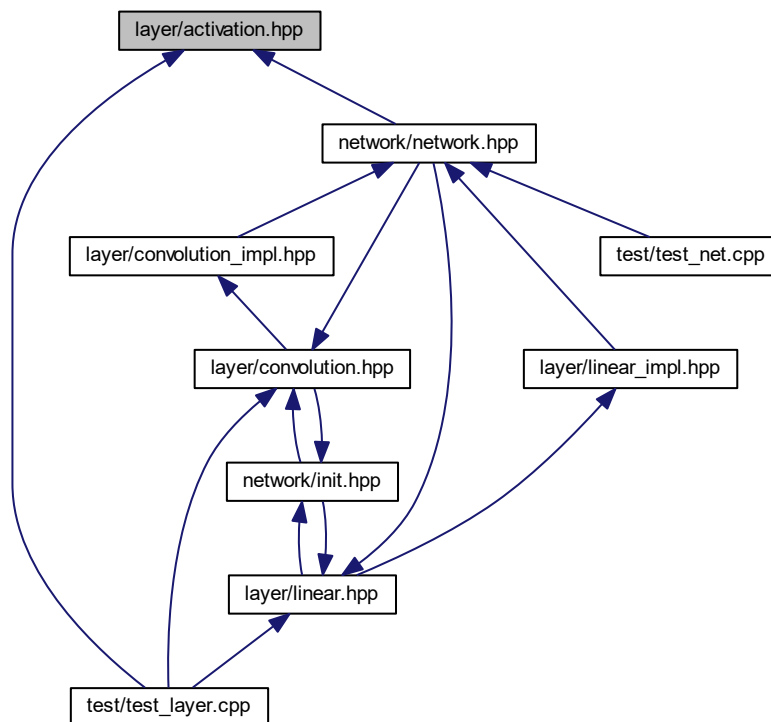
# File Documentation

### 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](mailto:lirj@shanghaitech.edu.cn)), YiFan Cao( [caoyf@shanghaitech.edu.cn](mailto:caoyf@shanghaitech.edu.cn)), Yan←  
Peng Hu( [huyf@shanghaitech.edu.cn](mailto:huyf@shanghaitech.edu.cn))

#### Version

1.6.0

#### Date

2019-05-30

#### Copyright

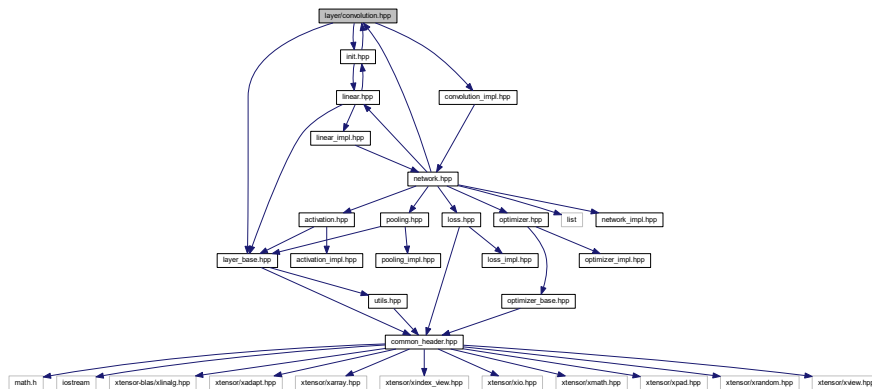
Copyright (c) 2019



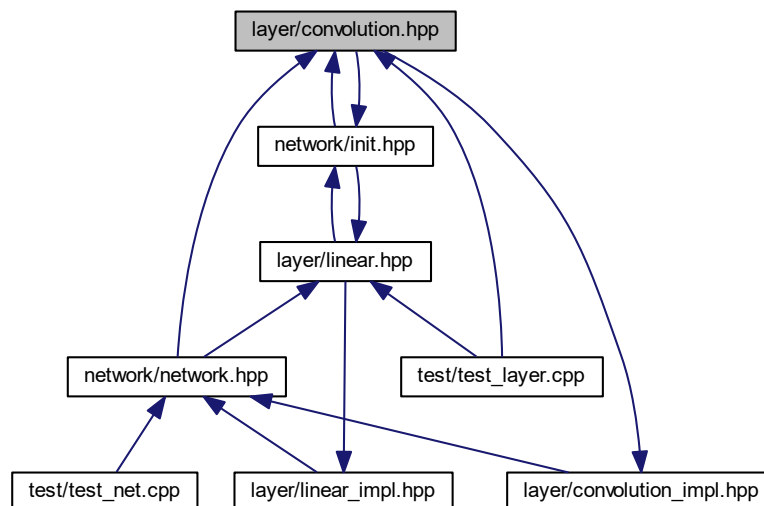
## 7.3 layer/convolution.hpp File Reference

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( [lijrj@shanghaitech.edu.cn](mailto:lijrj@shanghaitech.edu.cn)), YiFan Cao( [caoyf@shanghaitech.edu.cn](mailto:caoyf@shanghaitech.edu.cn)), Yan←  
Peng Hu( [huyup@shanghaitech.edu.cn](mailto:huyup@shanghaitech.edu.cn))

#### Version

1.6.0

#### Date

2019-05-30

#### Copyright

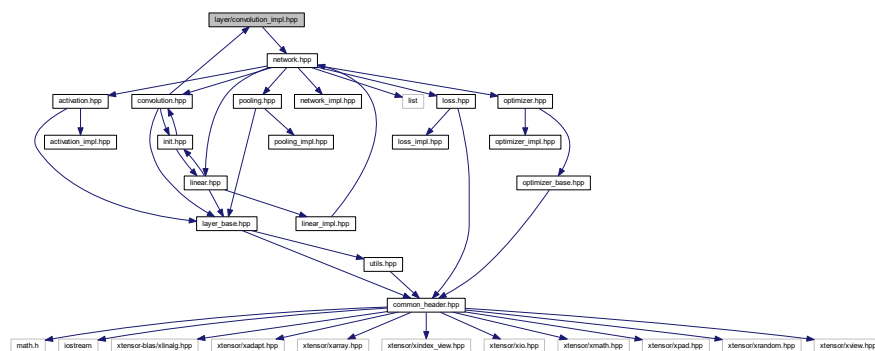
Copyright (c) 2019

## 7.4 layer/convolution\_impl.hpp File Reference

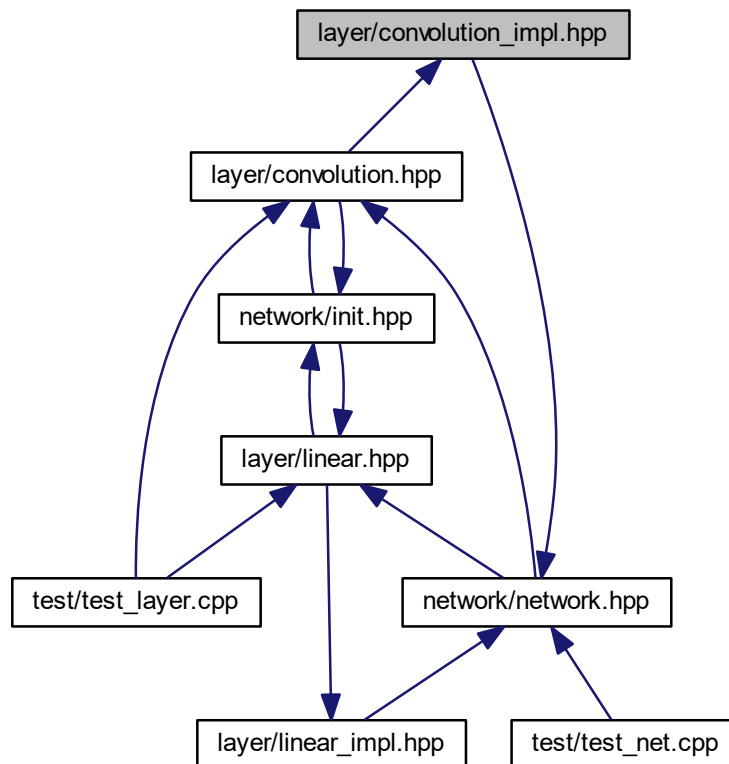
implementation for the convolution

```
#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](mailto:lirj@shanghaitech.edu.cn)), YiFan Cao( [caoyf@shanghaitech.edu.cn](mailto:caoyf@shanghaitech.edu.cn)), Yan↔  
Peng Hu( [huyf@shanghaitech.edu.cn](mailto:huyf@shanghaitech.edu.cn))

#### Version

1.6.0

#### Date

2019-05-30

#### Copyright

Copyright (c) 2019



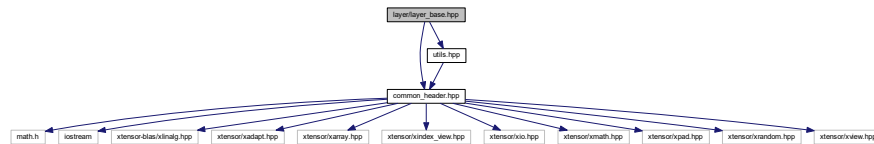
## 7.5 layer/layer\_base.hpp File Reference

the attribute of the base of the layer

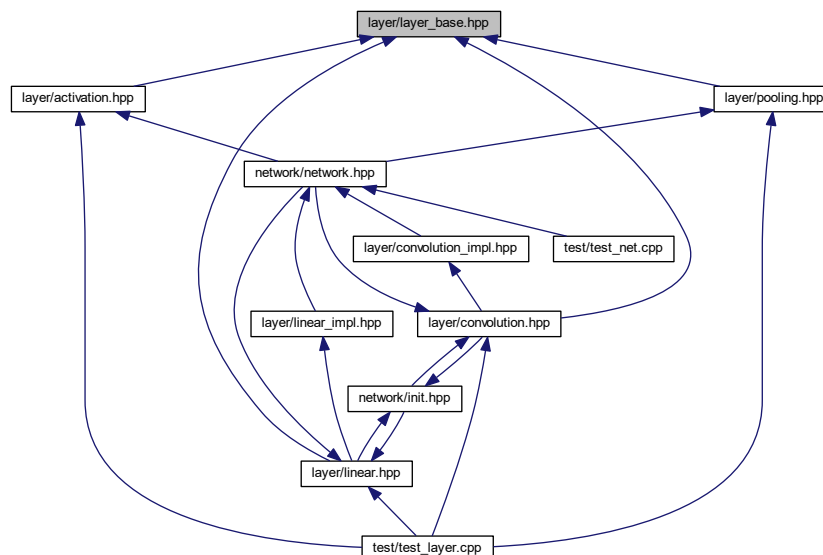
```
#include "common_header.hpp"
```

```
#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](mailto:lirj@shanghaitech.edu.cn)), YiFan Cao( [caoyf@shanghaitech.edu.cn](mailto:caoyf@shanghaitech.edu.cn)), Yan←  
Peng Hu( [huyf@shanghaitech.edu.cn](mailto:huyf@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	

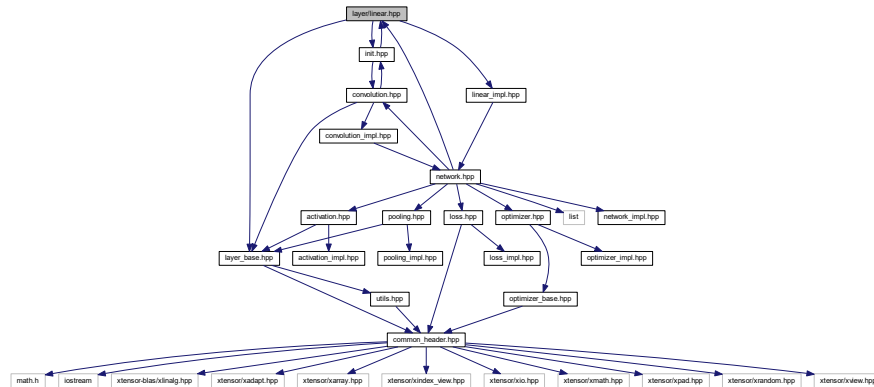
## 7.6 layer/linear.hpp File Reference

the linear of the header file

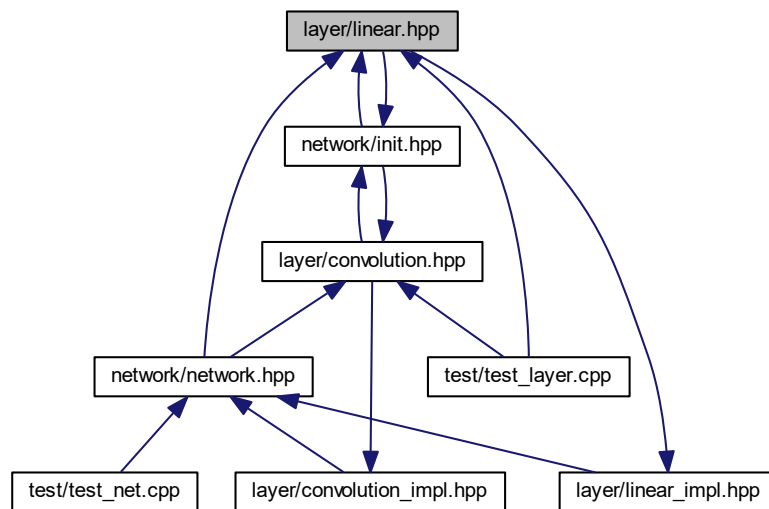
```
#include "init.hpp"  
#include "layer_base.hpp"
```

```
#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](mailto:lirj@shanghaitech.edu.cn)), YiFan Cao( [caoyf@shanghaitech.edu.cn](mailto:caoyf@shanghaitech.edu.cn)), Yan↔  
Peng Hu( [huyyp@shanghaitech.edu.cn](mailto:huyyp@shanghaitech.edu.cn))

**Version**

1.6.0

**Date**

2019-05-30

**Copyright**

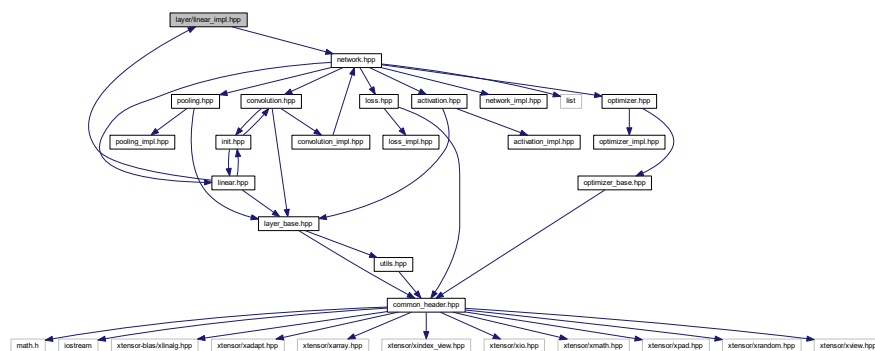
Copyright (c) 2019

## 7.7 layer/linear\_impl.hpp File Reference

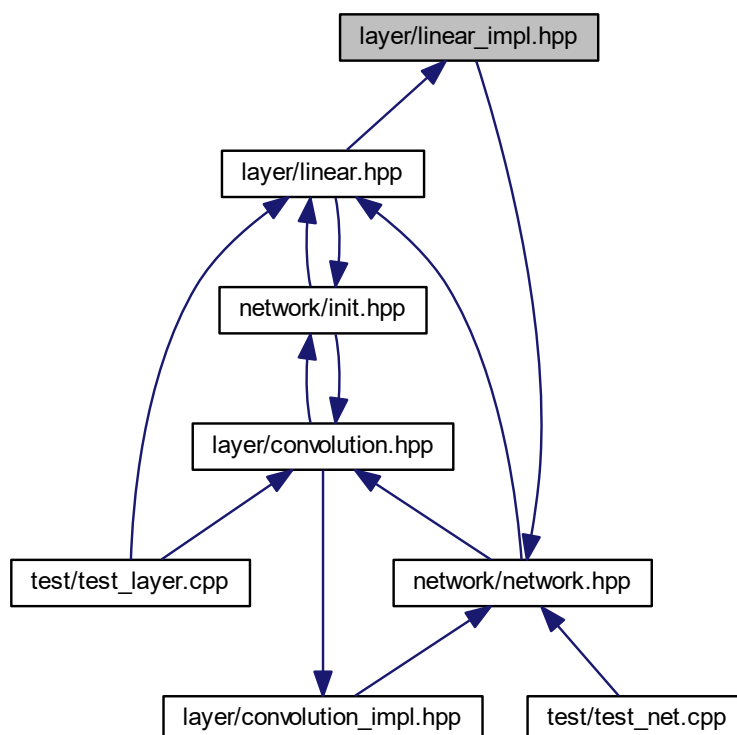
implementation of the linear & forward &backword

```
#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](mailto:lirj@shanghaitech.edu.cn)), YiFan Gao( [caoyf@shanghaitech.edu.cn](mailto:caoyf@shanghaitech.edu.cn)), Yan<→  
Peng Hu( [huyf@shanghaitech.edu.cn](mailto:huyf@shanghaitech.edu.cn))

#### Version

1.6.0

#### Date

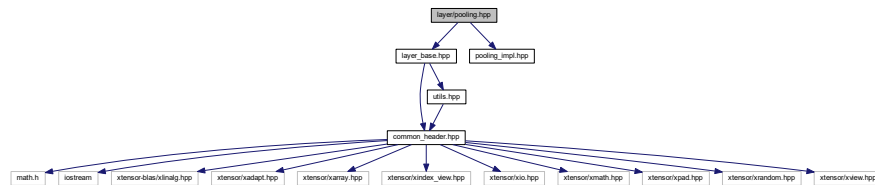
2019-05-30

#### Copyright

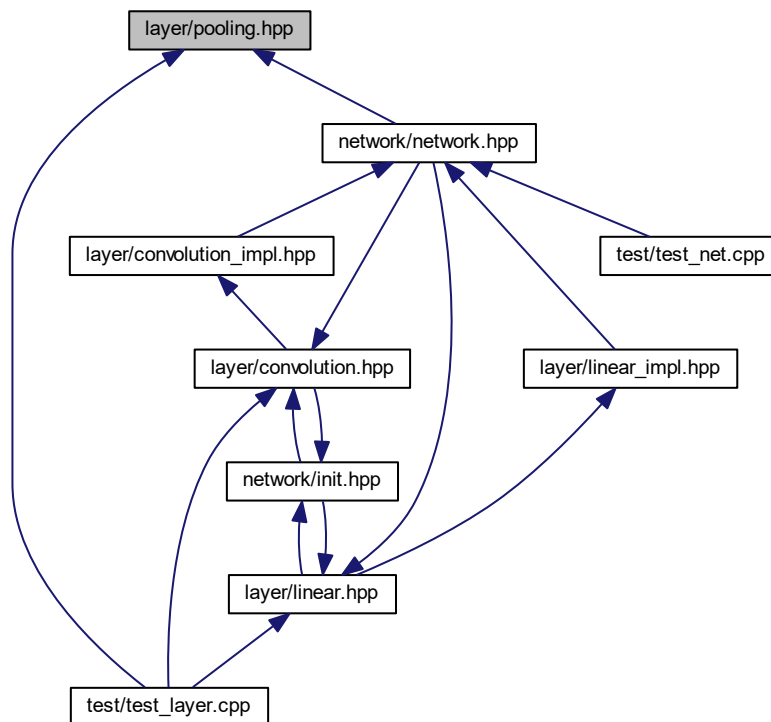
Copyright (c) 2019

## 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](mailto:lirj@shanghaitech.edu.cn)), YiFan Cao( [caoyf@shanghaitech.edu.cn](mailto:caoyf@shanghaitech.edu.cn)), Yan←  
Peng Hu( [huyf@shanghaitech.edu.cn](mailto:huyf@shanghaitech.edu.cn))

#### Version

1.6.0

#### Date

2019-05-30

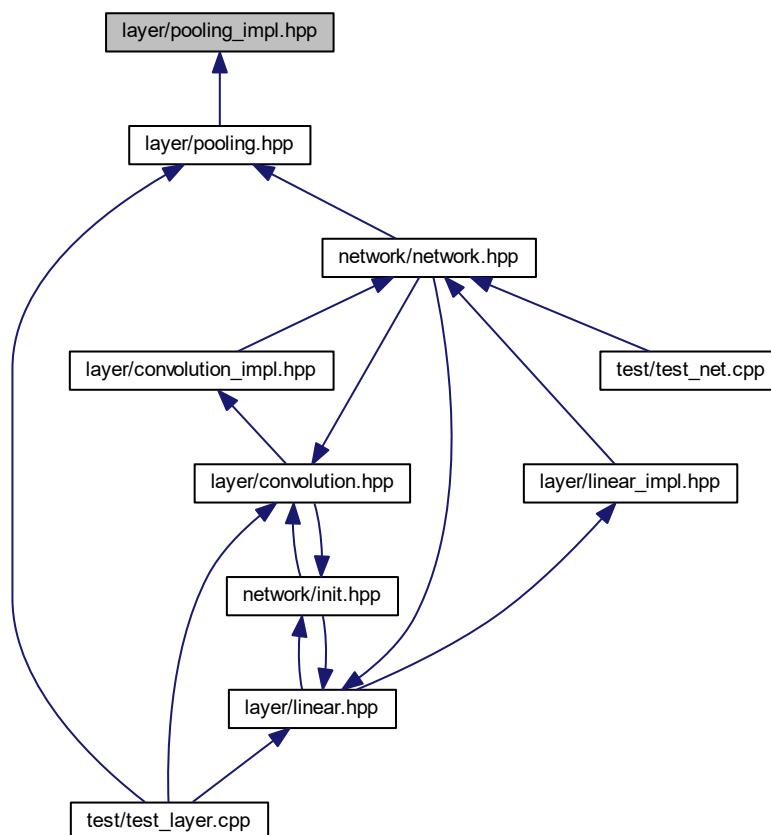
#### Copyright

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

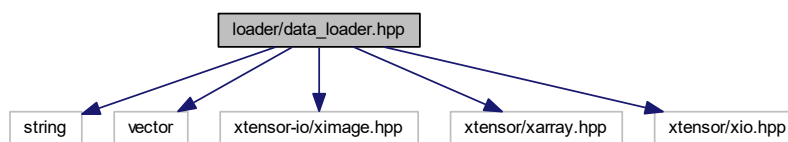
head file

## 7.10 loader/data\_loader.hpp File Reference

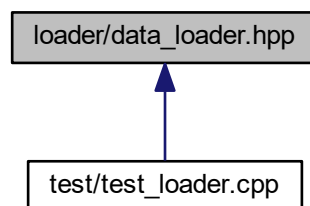
[data\\_loader.hpp](#)

```
#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_ )
```

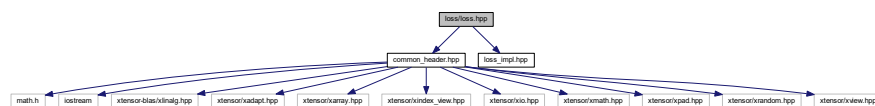
#### 7.10.2.2 load\_images()

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

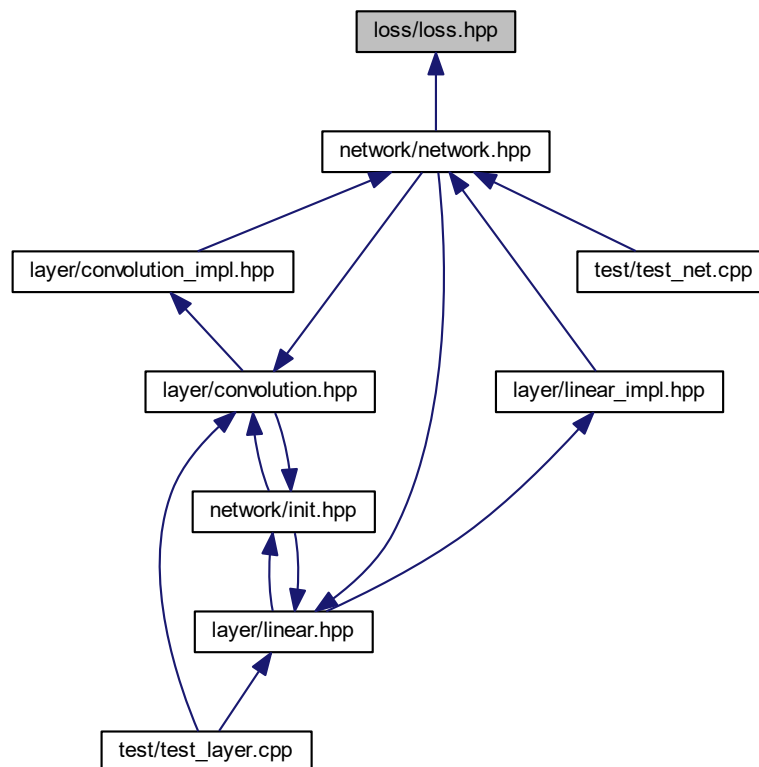
## 7.11 loader/model\_loader.hpp File Reference

## 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](#)

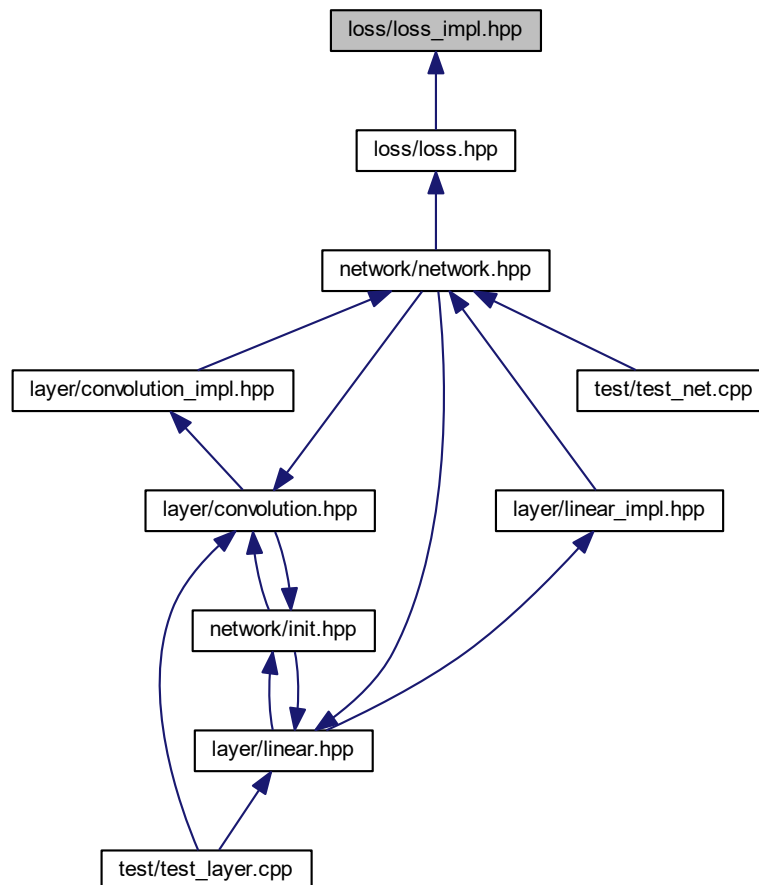
#### Enumerator

<a href="#">CROSS_ENTROPY</a>	
-------------------------------	--

## 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"

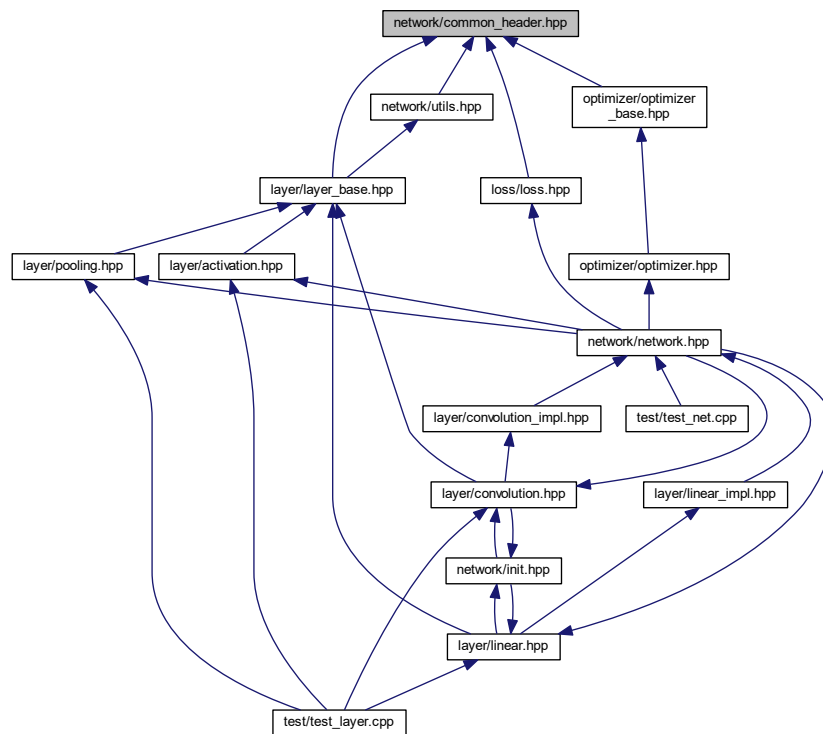
```

```
#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:

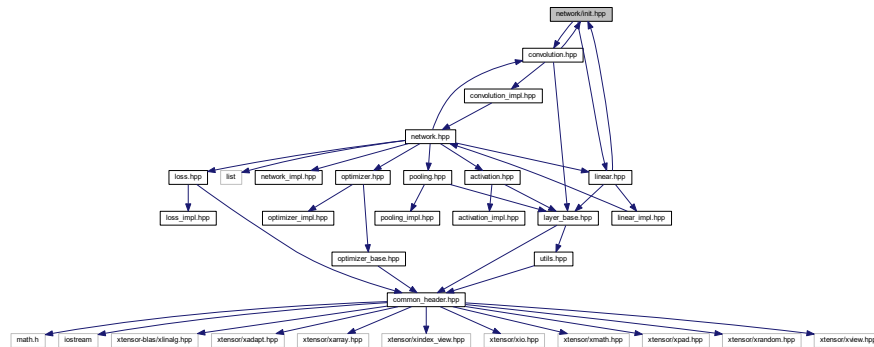


## 7.15 network/init.hpp File Reference

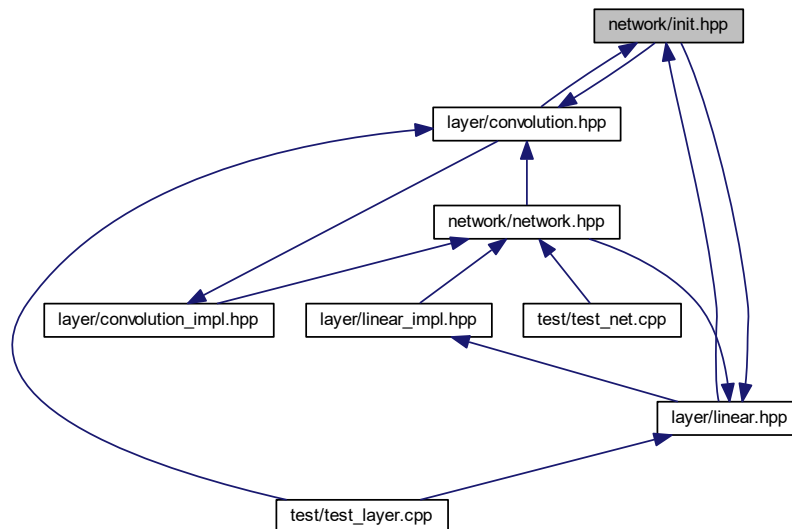
Init the network. Containing two functions: kaiming\_normal and kaiming\_uniform.

```
#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 functions: `kaiming_normal` and `kaiming_uniform`.

**Author**

RuiJian Li( [lirj@shanghaitech.edu.cn](mailto:lirj@shanghaitech.edu.cn)), YiFan Cao( [caoyf@shanghaitech.edu.cn](mailto:caoyf@shanghaitech.edu.cn)), Yan←  
Peng Hu( [huyf@shanghaitech.edu.cn](mailto:huyf@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**

<i>T</i>	
----------	--

**Parameters**

<i>layer</i>	
<i>mode</i>	

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 (
```

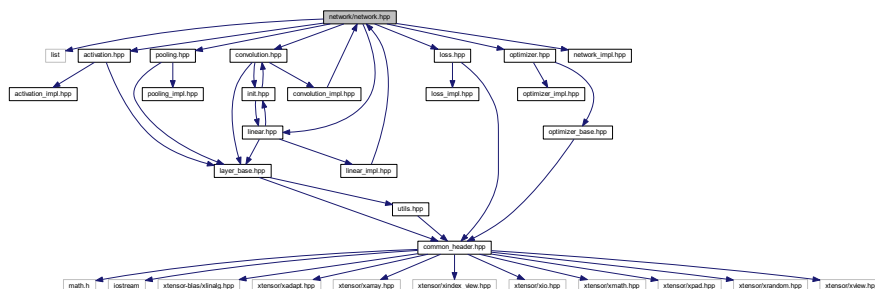
kaiming uniform distribution

$T$	
-----	--

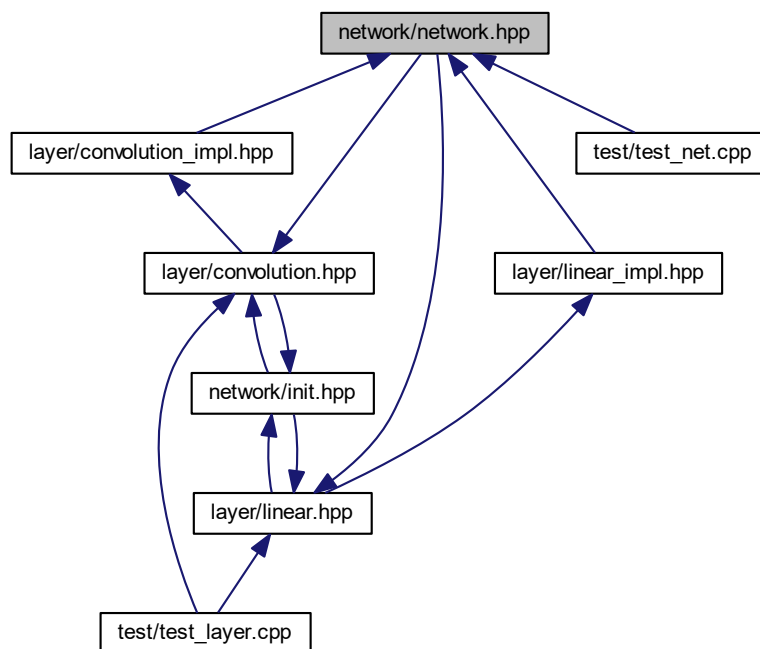
<i>layer</i>	
<i>mode</i>	

## 7.16 network/network.hpp File Reference

Include dependency graph for network.hpp:



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



## Classes

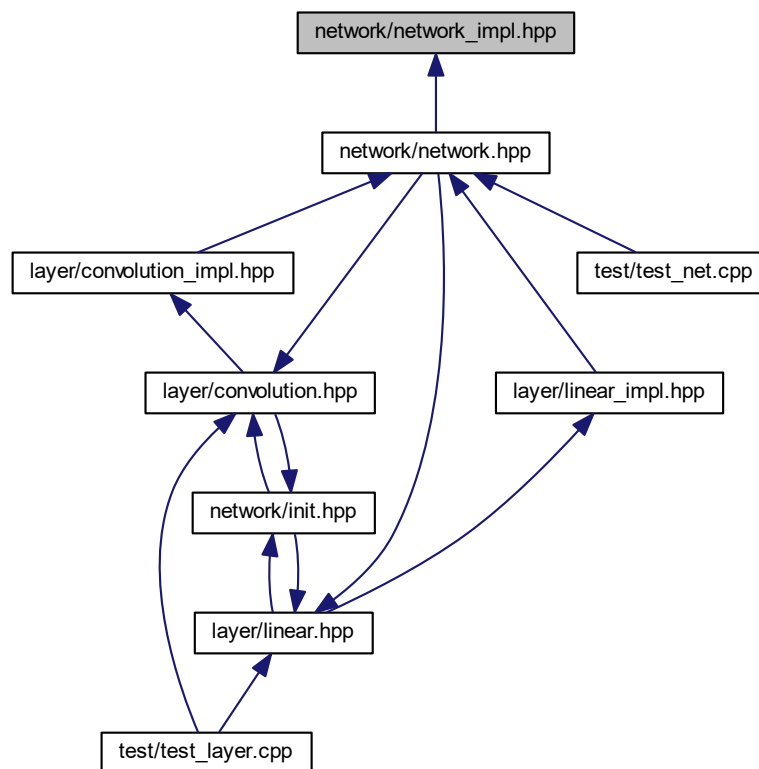
- class [Network< T >](#)

*the class of network*



## 7.17 network/network\_impl.hpp File Reference

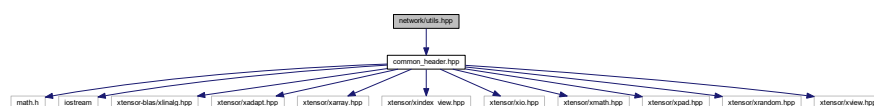
This graph shows which files directly or indirectly include this file:



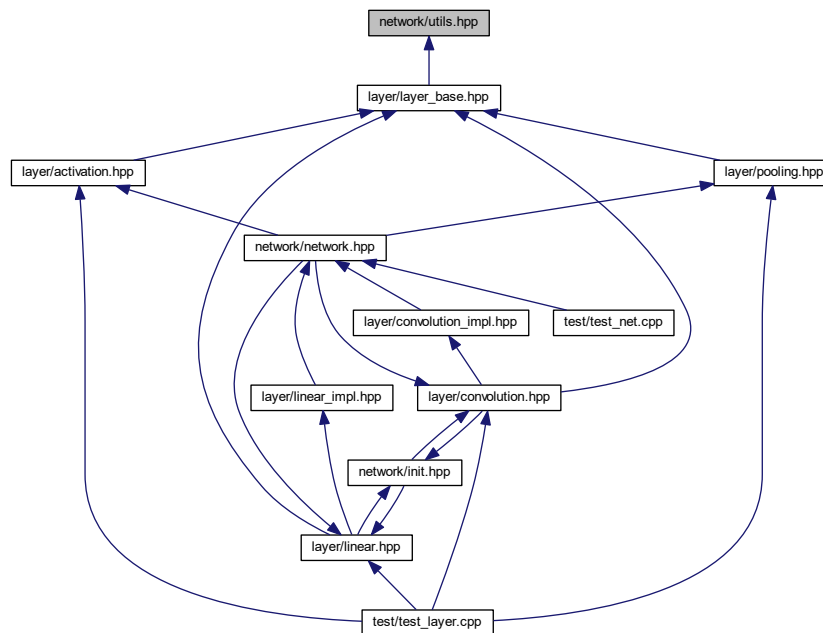
## 7.18 network/utils.hpp File Reference

```
#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)`

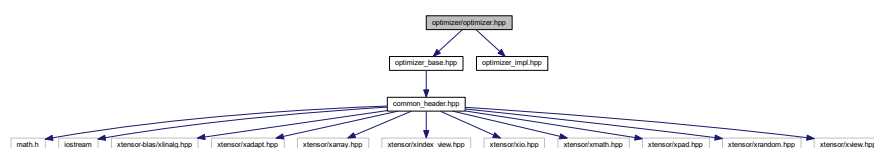
### 7.18.1 Function Documentation

#### 7.18.1.1 cout\_shape()

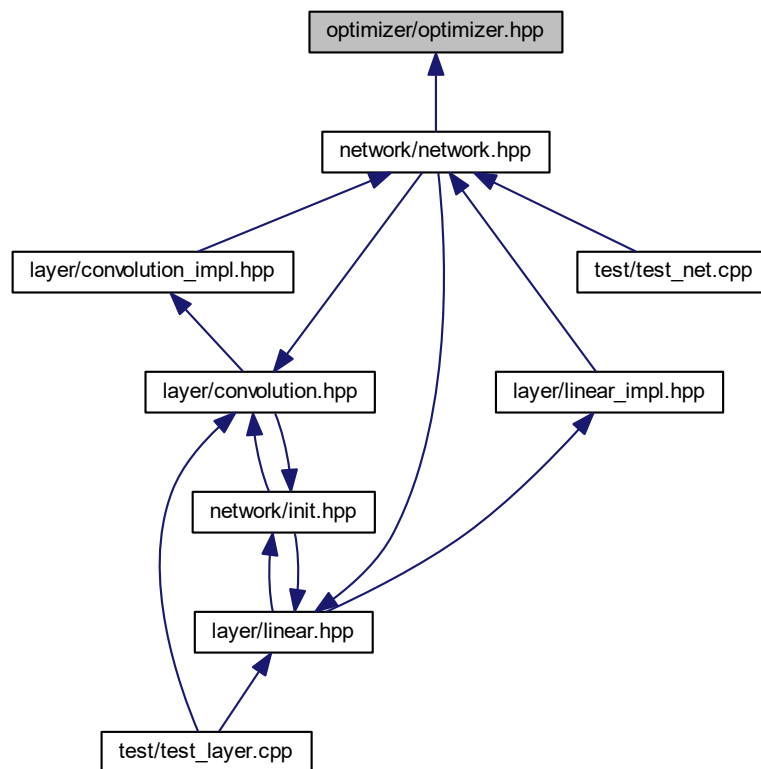
```
template<typename T >
void cout_shape (
    const xt::xarray< T > & matrix )
```

## 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 >`

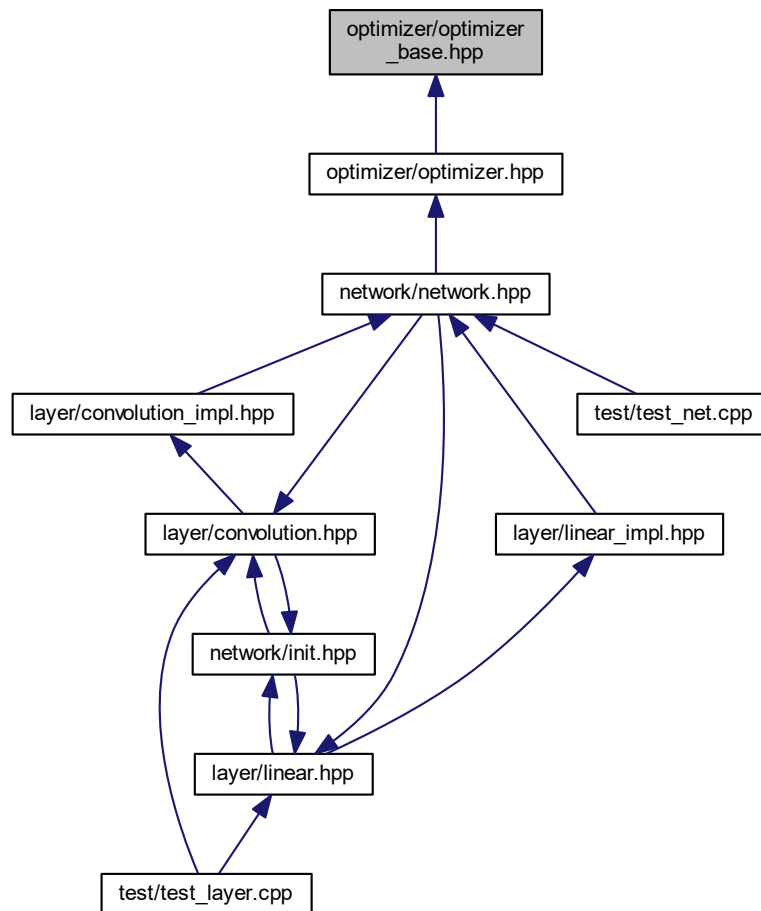
## 7.20 optimizer/optimizer\_base.hpp File Reference

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

Include dependency graph for `optimizer_base.hpp`:



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

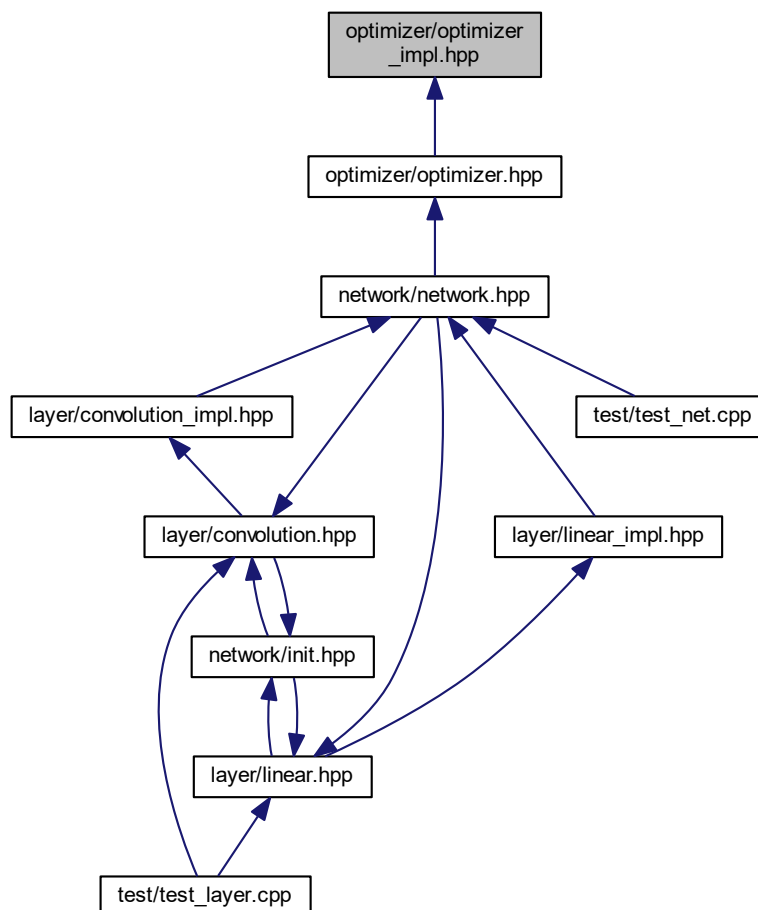


## Classes

- 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

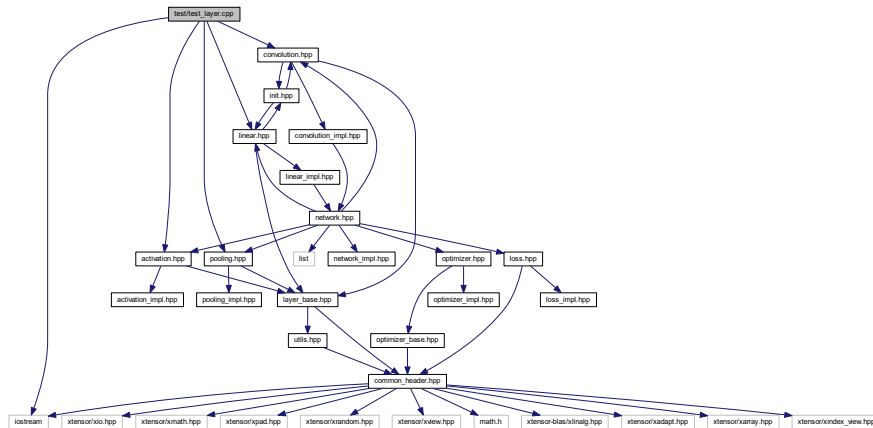
## 7.23 test/test\_layer.cpp File Reference

```

#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

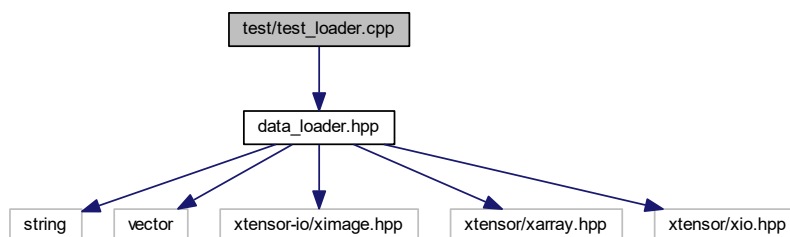
#### 7.23.1.1 main()

```
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:

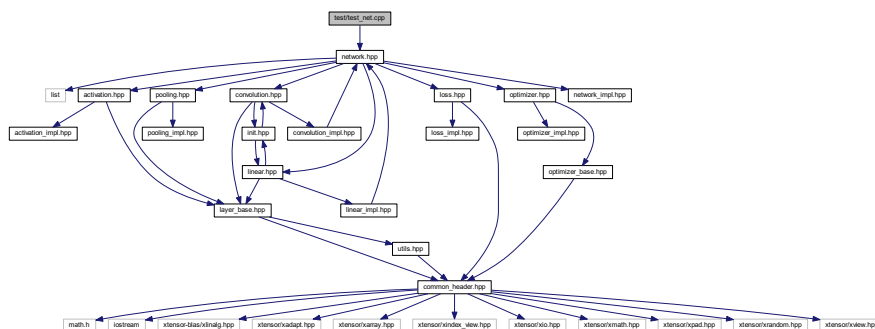


- int **main** (int argc, char \*\*argv)

### 7.24.1.1 main()

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

```
#include "network.hpp"
Include dependency graph for test_net.cpp:
```



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

### 7.25.1.1 main()

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





# Index

- ~Conv2d
  - Conv2d< T >, [13](#)
- ~Layer
  - Layer< T >, [18](#)
- ~Linear
  - Linear< T >, [25](#)
- ~Loss
  - Loss< T >, [30](#)
- ~MaxPool2d
  - MaxPool2d< T >, [33](#)
- ~Network
  - Network< T >, [38](#)
- ~Optimizer
  - Optimizer< T >, [41](#)
- ~ReLU
  - ReLU< T >, [45](#)
- ~SGD
  - SGD< T >, [48](#)
- ACT
  - layer\_base.hpp, [58](#)
- b\_
  - Layer< T >, [21](#)
- backward
  - Conv2d< T >, [14](#)
  - Layer< T >, [18](#)
  - Linear< T >, [25](#)
  - MaxPool2d< T >, [34](#)
  - Network< T >, [38](#)
  - ReLU< T >, [45](#)
- bias\_shape
  - Layer< T >, [19](#)
- CONV
  - layer\_base.hpp, [58](#)
- Conv2d
  - Conv2d< T >, [13](#)
- Conv2d< T >, [11](#)
  - ~Conv2d, [13](#)
  - backward, [14](#)
  - Conv2d, [13](#)
  - forward, [14](#)
  - get\_fan, [15](#)
  - in\_channels\_, [15](#)
  - kernel\_size\_, [15](#)
  - Matrix, [12](#)
  - out\_channels\_, [16](#)
  - padding\_, [16](#)
  - Shape, [12](#)
  - stride\_, [16](#)
- cout\_shape
  - utils.hpp, [74](#)
- CROSS\_ENTROPY
  - loss.hpp, [66](#)
- CrossEntropyLoss
  - Loss< T >, [30](#)
- data\_loader.hpp
  - image\_normalize, [65](#)
  - load\_images, [65](#)
- db\_
  - Layer< T >, [21](#)
- din\_
  - Layer< T >, [21](#)
- dscores\_
  - Loss< T >, [31](#)
- dW\_
  - Layer< T >, [21](#)
- forward
  - Conv2d< T >, [14](#)
  - Layer< T >, [19](#)
  - Linear< T >, [26](#)
  - MaxPool2d< T >, [35](#)
  - Network< T >, [38](#)
  - ReLU< T >, [45](#)
- get\_fan
  - Conv2d< T >, [15](#)
  - Layer< T >, [19](#)
  - Linear< T >, [27](#)
- get\_grad
  - Loss< T >, [30](#)
- get\_optimizer
  - Network< T >, [38](#)
- get\_type
  - Layer< T >, [20](#)
  - Loss< T >, [31](#)
- image\_normalize
  - data\_loader.hpp, [65](#)
- in\_
  - Layer< T >, [22](#)
- in\_channels\_
  - Conv2d< T >, [15](#)
- in\_dims\_
  - Linear< T >, [27](#)
- in\_reshape\_
  - Linear< T >, [28](#)

- init.hpp
  - kaiming\_normal, 70
  - kaiming\_uniform, 70
- kaiming\_normal
  - init.hpp, 70
- kaiming\_uniform
  - init.hpp, 70
- kernel\_size\_
  - Conv2d< T >, 15
  - MaxPool2d< T >, 36
- Layer
  - Layer< T >, 18
- Layer< T >, 16
  - ~Layer, 18
  - b\_, 21
  - backward, 18
  - bias\_shape, 19
  - db\_, 21
  - din\_, 21
  - dW\_, 21
  - forward, 19
  - get\_fan, 19
  - get\_type, 20
  - in\_, 22
  - Layer, 18
  - layer\_type\_, 22
  - Matrix, 17
  - net\_, 22
  - set\_bias, 20
  - set\_network, 20
  - set\_weight, 20
  - Shape, 18
  - W\_, 22
  - weight\_shape, 21
- layer/activation.hpp, 51
- layer/activation\_impl.hpp, 53
- layer/convolution.hpp, 54
- layer/convolution\_impl.hpp, 55
- layer/layer\_base.hpp, 57
- layer/linear.hpp, 58
- layer/linear\_impl.hpp, 60
- layer/pooling.hpp, 62
- layer/pooling\_impl.hpp, 63
- layer\_base.hpp
  - ACT, 58
  - CONV, 58
  - LAYER\_TYPE, 58
  - LINEAR, 58
  - POOL, 58
- LAYER\_TYPE
  - layer\_base.hpp, 58
- layer\_type\_
  - Layer< T >, 22
- layers\_
  - Network< T >, 39
- LINEAR
  - layer\_base.hpp, 58
- Linear
  - Linear< T >, 24, 25
- Linear< T >, 23
  - ~Linear, 25
  - backward, 25
  - forward, 26
  - get\_fan, 27
  - in\_dims\_, 27
  - in\_reshape\_, 28
  - Linear, 24, 25
  - Matrix, 24
  - out\_dims\_, 28
  - Shape, 24
- load\_images
  - data\_loader.hpp, 65
- loader/data\_loader.hpp, 64
- loader/model\_loader.hpp, 65
- Loss
  - Loss< T >, 29
- Loss< T >, 28
  - ~Loss, 30
  - CrossEntropyLoss, 30
  - dscores\_, 31
  - get\_grad, 30
  - get\_type, 31
  - Loss, 29
  - loss\_type\_, 31
  - Matrix, 29
  - scores\_, 31
  - Shape, 29
- loss.hpp
  - CROSS\_ENTROPY, 66
  - LOSS\_TYPE, 66
- loss/loss.hpp, 65
- loss/loss\_impl.hpp, 67
- loss\_
  - Network< T >, 39
- LOSS\_TYPE
  - loss.hpp, 66
- loss\_type\_
  - Loss< T >, 31
- lr\_
  - Optimizer< T >, 42
- main
  - test\_layer.cpp, 78
  - test\_loader.cpp, 79
  - test\_net.cpp, 79
- Matrix
  - Conv2d< T >, 12
  - Layer< T >, 17
  - Linear< T >, 24
  - Loss< T >, 29
  - MaxPool2d< T >, 33
  - Network< T >, 37
  - Optimizer< T >, 41
  - ReLU< T >, 44
  - SGD< T >, 47
- MaxPool2d

- MaxPool2d< T >, 33, 34
- MaxPool2d< T >, 32
  - ~MaxPool2d, 33
  - backward, 34
  - forward, 35
  - kernel\_size\_, 36
  - Matrix, 33
  - MaxPool2d, 33, 34
  - padding\_, 36
  - Shape, 33
  - stride\_, 36
- momentum\_
  - SGD< T >, 49
- net\_
  - Layer< T >, 22
- Network
  - Network< T >, 37
- Network< T >, 36
  - ~Network, 38
  - backward, 38
  - forward, 38
  - get\_optimizer, 38
  - layers\_, 39
  - loss\_, 39
  - Matrix, 37
  - Network, 37
  - operator<<, 38, 39
  - optimizer\_, 40
  - predict, 39
  - set\_optimizer, 39
  - Shape, 37
- network/common\_header.hpp, 67
- network/init.hpp, 68
- network/network.hpp, 71
- network/network\_impl.hpp, 73
- network/Utils.hpp, 73
- operator<<
  - Network< T >, 38, 39
- Optimizer
  - Optimizer< T >, 41
- Optimizer< T >, 40
  - ~Optimizer, 41
  - lr\_, 42
  - Matrix, 41
  - Optimizer, 41
  - Shape, 41
  - update, 41
- optimizer/optimizer.hpp, 74
- optimizer/optimizer\_base.hpp, 75
- optimizer/optimizer\_impl.hpp, 77
- optimizer\_
  - Network< T >, 40
- out\_channels\_
  - Conv2d< T >, 16
- out\_dims\_
  - Linear< T >, 28
- padding\_
  - Conv2d< T >, 16
  - MaxPool2d< T >, 36
- POOL
  - layer\_base.hpp, 58
- predict
  - Network< T >, 39
- README.md, 77
- ReLU
  - ReLU< T >, 44
- ReLU< T >, 42
  - ~ReLU, 45
  - backward, 45
  - forward, 45
  - Matrix, 44
  - ReLU, 44
  - Shape, 44
- scores\_
  - Loss< T >, 31
- set\_bias
  - Layer< T >, 20
- set\_network
  - Layer< T >, 20
- set\_optimizer
  - Network< T >, 39
- set\_weight
  - Layer< T >, 20
- SGD
  - SGD< T >, 48
- SGD< T >, 46
  - ~SGD, 48
  - Matrix, 47
  - momentum\_, 49
  - SGD, 48
  - Shape, 48
  - update, 48
  - weight\_decay\_, 49
- Shape
  - Conv2d< T >, 12
  - Layer< T >, 18
  - Linear< T >, 24
  - Loss< T >, 29
  - MaxPool2d< T >, 33
  - Network< T >, 37
  - Optimizer< T >, 41
  - ReLU< T >, 44
  - SGD< T >, 48
- stride\_
  - Conv2d< T >, 16
  - MaxPool2d< T >, 36
- test/test\_layer.cpp, 77
- test/test\_loader.cpp, 78
- test/test\_net.cpp, 79
- test\_layer.cpp
  - main, 78
- test\_loader.cpp

- main, [79](#)
- test\_net.cpp
  - main, [79](#)
- update
  - Optimizer< T >, [41](#)
  - SGD< T >, [48](#)
- utils.hpp
  - cout\_shape, [74](#)
- W\_
  - Layer< T >, [22](#)
- weight\_decay\_
  - SGD< T >, [49](#)
- weight\_shape
  - Layer< T >, [21](#)