

ANN\_cpp

Generated by Doxygen 1.9.1



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 Class Documentation</b>	<b>3</b>
2.1 Layer Class Reference	3
2.1.1 Detailed Description	3
2.1.2 Constructor & Destructor Documentation	4
2.1.2.1 Layer() [1/2]	4
2.1.2.2 Layer() [2/2]	4
2.1.3 Member Function Documentation	4
2.1.3.1 getActivationFunction()	4
2.1.3.2 getNbInput()	5
2.1.3.3 getNbNodes()	5
2.1.3.4 getNode()	5
2.1.3.5 operator=()	5
2.1.3.6 processOutputs()	6
2.2 Network Class Reference	6
2.2.1 Detailed Description	7
2.2.2 Constructor & Destructor Documentation	7
2.2.2.1 Network() [1/3]	7
2.2.2.2 Network() [2/3]	7
2.2.2.3 Network() [3/3]	7
2.2.3 Member Function Documentation	8
2.2.3.1 getLayer()	8
2.2.3.2 getLayerSize()	8
2.2.3.3 getNumberLayers()	8
2.2.3.4 LoadNetwork()	9
2.2.3.5 operator=()	9
2.2.3.6 processOutputs()	9
2.2.3.7 SaveNetwork()	9
2.3 Node Class Reference	10
2.3.1 Detailed Description	10
2.3.2 Constructor & Destructor Documentation	11
2.3.2.1 Node()	11
2.3.3 Member Function Documentation	11
2.3.3.1 getActivationFunction()	11
2.3.3.2 getBias()	11
2.3.3.3 getNbInput()	12
2.3.3.4 getWeight()	12
2.3.3.5 operator=()	12
2.3.3.6 processOutputs()	12
2.3.4 Friends And Related Function Documentation	13

2.3.4.1 read . . . . .	13
2.3.4.2 write . . . . .	13

<b>Index</b>	<b>15</b>
--------------	-----------

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Layer</a>	Class <a href="#">Layer</a> . . . . .	<a href="#">3</a>
<a href="#">Network</a>	Class <a href="#">Network</a> . . . . .	<a href="#">6</a>
<a href="#">Node</a>	Class <a href="#">Node</a> . . . . .	<a href="#">10</a>



## Chapter 2

# Class Documentation

### 2.1 Layer Class Reference

Class [Layer](#).

```
#include <layer.hpp>
```

#### Public Member Functions

- [Layer](#) ()  
*Construct a new empty [Layer](#) object.*
- [Layer](#) (activationFunction function, int nbInput, int nbNodes)  
*Construct a new [Layer](#) object.*
- [Layer](#) & [operator=](#) (const [Layer](#) &)=default  
*Equal operator for the [Layer](#) object.*
- std::vector< double > [processOutputs](#) (std::vector< double > inputs)  
*process the outputs of this layer*
- int [getNbInput](#) ()  
*Get the NbInput object.*
- int [getNbNodes](#) ()  
*Get the NbNodes object.*
- [Node](#) [getNode](#) (int index)  
*Get the [Node](#) object at the position index.*
- activationFunction [getActivationFunction](#) ()  
*Get the Activation Function object.*

#### 2.1.1 Detailed Description

Class [Layer](#).

which implements all the behaviour of a layer

## 2.1.2 Constructor & Destructor Documentation

### 2.1.2.1 Layer() [1/2]

```
Layer::Layer ( )
```

Construct a new empty [Layer](#) object.

Activation function used by the nodes of this layer

### 2.1.2.2 Layer() [2/2]

```
Layer::Layer (
    activationFunction function,
    int nbInput,
    int nbNodes )
```

Construct a new [Layer](#) object.

#### Parameters

<i>function</i>	: activation function used by the nodes of this layer
<i>nbInput</i>	: number of inputs in this layer
<i>nbNodes</i>	: number of nodes in this layer

## 2.1.3 Member Function Documentation

### 2.1.3.1 getActivationFunction()

```
activationFunction Layer::getActivationFunction ( )
```

Get the Activation Function object.

#### Returns

activationFunction



### 2.1.3.2 getNbInput()

```
int Layer::getNbInput ( )
```

Get the NbInput object.

#### Returns

int

### 2.1.3.3 getNbNodes()

```
int Layer::getNbNodes ( )
```

Get the NbNodes object.

#### Returns

int

### 2.1.3.4 getNode()

```
Node Layer::getNode (
    int index )
```

Get the [Node](#) object at the position index.

#### Parameters

<i>index</i>	: the position of the node we want to access.
--------------	---

#### Returns

[Node](#)

### 2.1.3.5 operator=()

```
Layer& Layer::operator= (
    const Layer & ) [default]
```

Equal operator for the [Layer](#) object.

#### Returns

[Layer&](#)

### 2.1.3.6 processOutputs()

```
std::vector< double > Layer::processOutputs (
    std::vector< double > inputs )
```

process the outputs of this layer

#### Parameters

<i>inputs</i>	: vector of inputs
---------------	--------------------

#### Returns

std::vector<double> : vector of outputs

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

- /home/marc/Documents/1. Développement/4. C++/1. Neural Network/ANN\_cpp/src/layer.hpp
- /home/marc/Documents/1. Développement/4. C++/1. Neural Network/ANN\_cpp/src/layer.cpp

## 2.2 Network Class Reference

Class [Network](#).

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

### Public Member Functions

- [Network](#) ()  
*Construct a new empty [Network](#) object.*
- [Network](#) (std::vector< int > Size, std::vector< activationFunction > activationFunctions)  
*Construct a new [Network](#) object.*
- [Network](#) (std::vector< int > Size, activationFunction activationfunction)  
*Construct a new [Network](#) object.*
- [Network](#) & operator= (const [Network](#) &)=default  
*Equal operator for the [Network](#) object.*
- std::vector< double > [processOutputs](#) (std::vector< double > inputs)  
*Process of the network layer by layer.*
- int [getLayerSize](#) (int index)  
*Get the [Layer](#) Size object.*
- int [getNumberLayers](#) ()  
*Get the Number Layers object.*
- [Layer](#) [getLayer](#) (int index)  
*Get the [Layer](#) object.*
- void [LoadNetwork](#) (std::string path)  
*Load a network from a binary file situated on the path.*
- void [SaveNetwork](#) (std::string path)  
*Save the network to a binary file situated on the path.*

## 2.2.1 Detailed Description

Class [Network](#).

which implements all the behaviour of a [Network](#).

## 2.2.2 Constructor & Destructor Documentation

### 2.2.2.1 [Network\(\)](#) [1/3]

```
Network::Network ( )
```

Construct a new empty [Network](#) object.

Size of the differents layers

### 2.2.2.2 [Network\(\)](#) [2/3]

```
Network::Network (
    std::vector< int > Size,
    std::vector< activationFunction > activationFunctions )
```

Construct a new [Network](#) object.

#### Parameters

<i>Size</i>	: the differents sizes of the layers.
<i>activationFunctions</i>	: the differents activations function used by the layers.

### 2.2.2.3 [Network\(\)](#) [3/3]

```
Network::Network (
    std::vector< int > Size,
    activationFunction activationfunction )
```

Construct a new [Network](#) object.

#### Parameters

<i>Size</i>	: the differents sizes of the layers.
<i>activationfunction</i>	: the activation function used by all the layers.

## 2.2.3 Member Function Documentation

### 2.2.3.1 `getLayer()`

```
Layer Network::getLayer (
    int index )
```

Get the [Layer](#) object.

#### Parameters

<i>index</i>	
--------------	--

#### Returns

[Layer](#)

### 2.2.3.2 `getLayerSize()`

```
int Network::getLayerSize (
    int index )
```

Get the [Layer](#) Size object.

#### Parameters

<i>index</i>	
--------------	--

#### Returns

int

### 2.2.3.3 `getNumberLayers()`

```
int Network::getNumberLayers ( )
```

Get the Number Layers object.

#### Returns

int

### 2.2.3.4 LoadNetwork()

```
void Network::LoadNetwork (
    std::string path )
```

Load a network from a binary file situated on the path.

#### Parameters

<i>path</i>	: path to the network .bin file
-------------	---------------------------------

### 2.2.3.5 operator=()

```
Network& Network::operator= (
    const Network & ) [default]
```

Equal operator for the [Network](#) object.

#### Returns

[Network&](#)

### 2.2.3.6 processOutputs()

```
std::vector< double > Network::processOutputs (
    std::vector< double > inputs )
```

Process of the network layer by layer.

#### Parameters

<i>inputs</i>	: vector of inputs.
---------------	---------------------

#### Returns

`std::vector<double>` : vector of outputs.

### 2.2.3.7 SaveNetwork()

```
void Network::SaveNetwork (
    std::string path )
```

Save the network to a binary file situated on the path.

#### Parameters

<i>path</i>	: path to the newly created .bin file
-------------	---------------------------------------

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

- /home/marc/Documents/1. Développement/4. C++/1. Neural Network/ANN\_cpp/src/network.hpp
- /home/marc/Documents/1. Développement/4. C++/1. Neural Network/ANN\_cpp/src/network.cpp

## 2.3 Node Class Reference

Class [Node](#).

```
#include <node.hpp>
```

### Public Member Functions

- [Node](#) (activationFunction function, int nbIn)  
*Construct a new [Node](#) object.*
- [Node](#) ()  
*Empty constructor for [Node](#) object.*
- [Node](#) & [operator=](#) (const [Node](#) &)=default  
*Equal operator for the node object.*
- double [processOutputs](#) (std::vector< double > inputs)  
*Method to process the outputs of the node.*
- double [getWeight](#) (int index)  
*Get one of the weight.*
- double [getBias](#) ()  
*Get the Bias.*
- int [getNbInput](#) ()  
*Get the number of input.*
- activationFunction [getActivationFunction](#) ()  
*Get the Activation Function of the [Node](#).*

### Friends

- std::ostream & [write](#) (std::ostream &out, [Node](#) &obj)  
*Overload of the write function for the node object.*
- std::istream & [read](#) (std::istream &out, [Node](#) &obj)  
*Overload of the read function for the node object.*

### 2.3.1 Detailed Description

Class [Node](#).

Class which represent the behaviour of a [Node](#) in an Artificial Neural [Network](#).

## 2.3.2 Constructor & Destructor Documentation

### 2.3.2.1 Node()

```
Node::Node (
    activationFunction function,
    int nbIn )
```

Construct a new [Node](#) object.

Construct a new [Node](#) object by creating random weights and a random bias.

#### Parameters

<i>nbIn</i>	corresponds to the number of inputs of the created <a href="#">Node</a> .
-------------	---

## 2.3.3 Member Function Documentation

### 2.3.3.1 getActivationFunction()

```
activationFunction Node::getActivationFunction ( )
```

Get the Activation Function of the [Node](#).

#### Returns

activationFunction : the [Node](#) Activation Function.

### 2.3.3.2 getBias()

```
double Node::getBias ( )
```

Get the Bias.

#### Returns

double : the bias.

### 2.3.3.3 getNbInput()

```
int Node::getNbInput ( )
```

Get the number of input.

#### Returns

int : the number of input.

### 2.3.3.4 getWeight()

```
double Node::getWeight (
    int index )
```

Get one of the weight.

#### Parameters

<i>index</i>	of the weight we want.
--------------	------------------------

#### Returns

double : the weight.

### 2.3.3.5 operator=()

```
Node& Node::operator= (
    const Node & ) [default]
```

Equal operator for the node object.

#### Returns

the address [Node](#) of the left sided [Node](#) object.

### 2.3.3.6 processOutputs()

```
double Node::processOutputs (
    std::vector< double > inputs )
```

Method to process the outputs of the node.

Method which realise the calculation of the output by doing the dot product of the weights by the inputs. Then it add the bias and finally it use the Activation Function on the resulting scalar.



**Parameters**

<i>inputs</i>	: the inputs of the node.
---------------	---------------------------

**Returns**

double : the state of the node after the calculation.

## 2.3.4 Friends And Related Function Documentation

### 2.3.4.1 read

```
std::istream& read (
    std::istream & out,
    Node & obj ) [friend]
```

Overload of the read function for the node object.

**Parameters**

<i>out</i>	
<i>obj</i>	

**Returns**

std::istream&

### 2.3.4.2 write

```
std::ostream& write (
    std::ostream & out,
    Node & obj ) [friend]
```

Overload of the write function for the node object.

**Parameters**

<i>out</i>	
<i>obj</i>	

**Returns**

std::ostream&

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

- /home/marc/Documents/1. Développement/4. C++/1. Neural Network/ANN\_cpp/src/node.hpp
- /home/marc/Documents/1. Développement/4. C++/1. Neural Network/ANN\_cpp/src/node.cpp

# Index

- getActivationFunction
  - Layer, [4](#)
  - Node, [11](#)
- getBias
  - Node, [11](#)
- getLayer
  - Network, [8](#)
- getLayerSize
  - Network, [8](#)
- getNbInput
  - Layer, [4](#)
  - Node, [11](#)
- getNbNodes
  - Layer, [5](#)
- getNode
  - Layer, [5](#)
- getNumberLayers
  - Network, [8](#)
- getWeight
  - Node, [12](#)
- Layer, [3](#)
  - getActivationFunction, [4](#)
  - getNbInput, [4](#)
  - getNbNodes, [5](#)
  - getNode, [5](#)
  - Layer, [4](#)
  - operator=, [5](#)
  - processOutputs, [5](#)
- LoadNetwork
  - Network, [8](#)
- Network, [6](#)
  - getLayer, [8](#)
  - getLayerSize, [8](#)
  - getNumberLayers, [8](#)
  - LoadNetwork, [8](#)
  - Network, [7](#)
  - operator=, [9](#)
  - processOutputs, [9](#)
  - SaveNetwork, [9](#)
- Node, [10](#)
  - getActivationFunction, [11](#)
  - getBias, [11](#)
  - getNbInput, [11](#)
  - getWeight, [12](#)
  - Node, [11](#)
  - operator=, [12](#)
  - processOutputs, [12](#)
  - read, [13](#)
  - write, [13](#)
- operator=
  - Layer, [5](#)
  - Network, [9](#)
  - Node, [12](#)
- processOutputs
  - Layer, [5](#)
  - Network, [9](#)
  - Node, [12](#)
- read
  - Node, [13](#)
- SaveNetwork
  - Network, [9](#)
- write
  - Node, [13](#)