

# Photonic Artificial Neuromorphic Networks: All-Optical Activation Function

Davide Bazzanella

Department of Physics, University of Trento

February 4, 2018

# Contents

|   |           |
|---|-----------|
| <b>Introduction</b>                                       | <b>1</b>  |
| <b>1 Artificial Neural Networks</b>                       | <b>5</b>  |
| 1.0.1 History . . . . .                                   | 5         |
| 1.0.2 Comparison with conventional computers . . . . .    | 6         |
| 1.1 Basis of Neural Networks . . . . .                    | 6         |
| 1.2 Working Principles of ANNs . . . . .                  | 8         |
| 1.2.1 Learning Process . . . . .                          | 8         |
| 1.2.2 Validation Process . . . . .                        | 10        |
| 1.2.3 Testing Process . . . . .                           | 10        |
| 1.2.4 Datasets . . . . .                                  | 10        |
| 1.3 Feedforward NN . . . . .                              | 10        |
| 1.3.1 Other Types of NNs . . . . .                        | 11        |
| 1.4 Real-Life Examples . . . . .                          | 13        |
| 1.5 ANN Simulation . . . . .                              | 13        |
| 1.5.1 PyTorch . . . . .                                   | 13        |
| <b>2 Integrated Photonics</b>                             | <b>15</b> |
| 2.0.1 Silicon Photonics ? . . . . .                       | 15        |
| 2.1 Guided-wave photonics . . . . .                       | 15        |
| 2.1.1 Waveguides . . . . .                                | 15        |
| 2.1.2 Light coupling . . . . .                            | 18        |
| 2.1.3 Microring Optical Cavity . . . . .                  | 18        |
| 2.1.4 Nonlinear Perturbations . . . . .                   | 19        |
| 2.2 Integrated photonics applied to ANNs . . . . .        | 19        |
| 2.2.1 Weighted sum of inputs . . . . .                    | 20        |
| 2.2.2 Nonlinear Activation Function . . . . .             | 20        |
| 2.2.3 Simulations . . . . .                               | 20        |
| <b>3 Samples, setup and experiments</b>                   | <b>21</b> |
| 3.1 The samples . . . . .                                 | 21        |
| 3.2 Setup . . . . .                                       | 21        |
| 3.3 Characterization of the Activation Function . . . . . | 21        |
| 3.4 Test of a Trained ANN . . . . .                       | 22        |
| <b>Conclusions</b>  | <b>23</b> |
| <b>Bibliography</b>                                       | <b>25</b> |
| <b>Index</b>  | <b>27</b> |

**Aknowledgements****29**

# Introduction

## The research field

Research in the field of artificial neural networks has become more and more popular in the last few decades as the computing power available to the average institution increased further. As a matter of fact, in the last two decades the calculation power which once belonged only to bulky, power hungry, and very expensive supercomputers, gradually became possible also for relatively more compact, efficient, and definitely cheaper servers and workstations.

This trend is dictated by the fact that artificial neural networks are usually simulated on conventional computers, which are on the other hand based on the von Neumann, or Princeton, architecture. Implementations of this general purpose architecture are able to do any logical computation with a series of instructions. However the downside of this suitability to generic computations is that it is often difficult to parallelize single operations and therefore certain tasks are inherently inefficient, both in energy and in time. Moreover another problem is that it requires explicit programming for each singular task

The formalization of the von Neumann architecture dates back to 1945 and is as old as the first attempt in artificial neural networks. Shortly after, research and industry efforts focused only on the development of the Princeton architecture, choosing it de facto as the primary design for computing devices.

In the recent years, help came from acceleration units known as Graphic Processing Units (GPU), which were developed for a completely different objective. These GPUs, which are nowadays being designed specifically to accelerate calculations of artificial neural network simulations, are able to carry out a restricted set of instruction compared to CPUs, but are much more efficient both in power and in time, because of their parallel execution.

## Today's software implementations

Between the most famous examples of simulated neural network there certainly are the older Watson supercomputer from IBM and the more recent AlphaGo artificial intelligence. Both of them arose to popularity because they defeated us humans to our own games. Specifically the effort of Watson was aimed at prevailing the most strongest human contestants at *Jeopardy!*, an American television game, and succeeded in 2011. Watson was powered by a IBM supercomputer and used 0.22 MW of power. AlphaGO, nevertheless, is a deep neural network that in 2016, through reinforced learning algorithms, defeated Lee Sedol, considered the most formidable master of Go, a complicated board game.

Another interesting case is given by SpiNNaker, of the University of Manchester. It is composed by  $10^4$  neurons connected by more than  $10^7$  synapses, and is simulated on over 65 thousand 18-core ARM processors connected together. Unlike the other two, its main goals are to simulate several neural mechanisms, such as the operation of visual cortex. The SpiNNaker system is part of the Human Brain Project (HBP), funded by the European Union.[1]

Similarly to SpiNNaker, another effort in gaining better understanding of the human brain is *The Blue Brain Project*, which is leaded by École Polytechnique Fédérale de Lausanne in collaboration with over one hundred other international research institutions and it is funded by the European Union. This project is simulated on IBM Blue Gene supercomputers, each

fitted with more than 100 thousand processors and consuming several MW of power.

### **Present objectives and future trends**

We can distinguish two principal objectives in this research field. The first one is a technological objective and is that of developing a more powerful computational instrument. The second objective is instead a topic of more fundamental research: to use the tools of artificial neural networks to better understand biological neural networks such as our brain.

If the comprehension of the complex mechanism at the base of our brain has always been a very arduous assignment, the tasks that artificial neural network are facing become increasingly difficult while time passes. Whereas in the past years public and private research groups trained themselves with "easier" problems, such as television or board games, today they are concentrating their efforts on more challenging goals. Probably the most clear example of this tendency is given by the research on autonomous driving, where artificial neural networks are used primarily for real-time processing of data from several sensors.

A great number of businesses formed in recent years around these problems and many other will probably do the same for similar areas. It is very likely that to achieve their objectives, even more powerful and complex artificial networks will be required. Therefore more processors will be developed and put together to achieve impressive computing power. However, one cannot expect to increase indefinitely the power, without significant improvements in efficiency. Hence alternatives are sought.

### **Overview of current alternatives to simulations**

Apart from the vast majority of the research on neural networks, which is being carried out with simulations on powerful processors, a smaller portion of research is focused on building neural network physically. Great efforts are made nowadays by many research institutions and companies to develop original computing architectures that allow artificial neural networks to run physically, instead of being simulated on conventional computers.

The reason why considerable improvements can be expected is that the path of designing hardware devices ad hoc for neural network computation has not yet been covered comprehensively. Major performance or efficiency enhancements are coming from the fact that those new architectures are conceived with parallel execution of specific operations in mind. This, however, makes them inevitably less adaptable to changes, unlike software simulations on conventional computers, which can be modified just by changing few lines of code. A good example is given by size of the network, which in simulation is easily altered, even of orders of magnitude, while in hardware implementations it depends strictly by the architecture features and scalability.

Many of these use the electronics framework, mainly because of the many benefits of the mature silicon technology and its CMOS-compatible production chain. Specifically footprint and efficiency of CMOS devices has not yet stopped to grow since its discovery, in the past century. However, a relatively small portion of these works also inquired simpler network fabrication with organic electronic materials, achieving remarkable results.

Among the most interesting projects which uses electronics there are TrueNorth of IBM, Neurogrid of Stanford University, and BrainScaleS designed in Heidelberg and Dresden. The former is part of a long-term research project funded by DARPA and has the aim of building an artificial neural network with the capabilities of brains of small mammals, such as cats or mice. In 2014 IBM showed a chip, named TrueNorth, containing 1 million artificial neurons and 256 millions synapses. This network, relying on more than 5 billion transistors organized on an area of  $4.3\text{ cm}^2$ , consumed only 60 mW. The researcher also proved that connecting 48 of this chips together they obtained the equivalent of the brain of a mouse.

Similarly, the intention behind Neurogrid chip is to explore numerous hypothesis concerning mammalian brains, specifically about the inner mechanisms of operation of the cerebral cortex. Stanford's implementation reaches the line of 1 million neurons, while only requiring 5 W of power consumption. Moreover the communication lines between single neurons, the synapses, are provided by FPGAs and banks of SRAM.

Finally the BrainScaleS is built by 20 silicon wafers, each containing 200,000 biologically realistic neurons and  $50 \times 10^6$  plastic synapses. Likewise SpiNNaker is funded by the European Union and it is part of the Human Brain Project. A particularity of this system is that does not execute pre-programmed code but evolves according to the physical properties of the devices.[1]

Besides electronics, other kind of physical implementations are attempted from research teams. Probably the most intriguing example is given by researches in photonics, which attempts to match electronics performance by overcoming its intrinsically weaknesses, such as power consumption, speed, and intrinsic compatibility with parallel computing. Photonics has yet to reach achievements comparable to electronics, mainly due to the embryonic state of its technology.

## My work

My work in this thesis has two related objectives. The primary aim is to implement a proof of concept for a fundamental component of artificial neural networks: the neuron's activation function. I plan to do so by employing the framework made available by integrated photonics. The second intent is to bring together two until now distant fields of research: physics and information technology, specifically photonics and machine learning. In fact generating knowledge transversal to the two research fields is at least as much important as the primary objective.

The idea at the base of our physical implementation is to employ integrated optical devices, which have been manufactured for other projects, to create a proof of concept for the activation function. Therefore in this work I will study a particular device, a microring resonator, and characterize its response to different inputs, for several working conditions. This device can process information because it has, under certain circumstances, a nonlinear response function to its inputs. Moreover, the same device might be used in another important part of artificial neural network nodes: the weighted sum. With this two parts together one could claim to have built a rudimentary artificial neuromorphic network, the *perceptron* (see 1.3). However, since integrated structures able to carry out a weighted sum of optical signals have already been discussed in literature and it requires less effort to produce acceptable results, it will not be at the center of this study.

During the thesis I achieved two main results. I have demonstrated the use of optical bistability in microring cavities to produce an activation function I have demonstrated the use of the response of a microring optical cavity to produce an activation function by exploiting the optical bistability regime. In addition, with the aim of testing such response function, I simulated an artificial network with custom activation function by using standard software libraries used by ICT community to model neural networks.

## Structure of the thesis

The first chapter of this thesis is intended as an introduction to the world of artificial neural networks. After a brief historical summary, I describe the main aspects of neural networks and their working principles. Then I focus on the description of the simplest type of artificial networks, the feedforward network, and finally and overview on other kind of networks. Later, the chapter is concluded with an introductory description of the language Python and the library PyTorch, used in this work for artificial network simulations.

The second chapter introduces the physics which I will study. Again, after a short explanation on the basic devices used in the field, I will describe in depth the device studied for the task. Having clarified the physics underneath, I ought to explain how I am going to use this physics to implement (part of) an artificial neural network. I will address these topics both from the theoretical point of view and from the numerical one, with simulation of the specific (simplified) system.

The third chapter is dedicated to the description of the experimental work. At the beginning the problem of the selection of the sample is discussed, along with the definition of the device chosen. After, I characterize the different behaviors with detailed data. Finally, an explanation on how the device will be used in the neural network viewpoint and the corresponding tests of operation.

# Chapter 1

## Artificial Neural Networks

Artificial Neuromorphic Networks (ANNs) are computational systems which elaborate information in a way that is loosely inspired by the operation of biological neural networks (animal brains). Biological networks are superior in performance to computers and extremely more efficient in difficult tasks such as classification (e.g. image recognition) and prediction (e.g. pattern recognition). The underlying idea is to copy some of their mechanisms and exploit them for computational applications. These systems are intrinsically parallel in operation, do not require specific programming to operate, and modify their behavior through a learning process to improve their accuracy in a certain task.

Mathematically speaking, ANNs are a collection of nodes, each one of them elaborates the information and is somehow connected to the other. Artificial networks can be either simulated on computers or physically built on hardware designed ad hoc. At this time, ANNs are mainly implemented by simulations, however the technological progress is withheld by limitation in computing power and efficiency. Training a complex artificial neural network with computers at the state of art might take even weeks. Thus, with the aim of improving performance from simulated networks, research on hardware architectures is surging: digital, analog, electrical, and optical devices are being developed.

### 1.0.1 History

It is widely acknowledged that the opening work of this research field was made in 1943 by Warren McCulloch and Walter Pitts, a neurophysiologist and a mathematician respectively. In their research work, they described the operating mechanism of biological neurons by modeling a simple electronic circuit [2].

The following important work was made by Donald O. Hebb, who hypothesized the neural plasticity in 1949 [3]. He pointed out the fact that neural pathways are strengthened each time they are used, a concept known as *Hebbian learning*.

During the late 1950s and the early 1960s, as computers became more powerful, many promising works were published. Some simulated operation of artificial networks on calculators, e.g. Farley and Clark [4] and Rochester [5]. Other produced circuitry that implemented on hardware such networks, e.g. Rosenblatt [6], [7]. However, despite the early successes of neural networks, the traditional computing architecture (von Neumann architecture) was chosen as the preferred computing architecture.

The reason why this happened, probably, was due to several concurring facts. First of all, in the same time period (1969) a research paper by Minsky and Papert [8], which identified two important problems. The basic perceptron was not able to execute the exclusive-or (XOR) operation, unlike the logical circuits at the base of von Neumann architecture. Moreover, the research stated the fact that more complex networks such as multiple- or deep-layered networks were not possible (at that time) because of the lack of adequate processing power.



In addition to this research, the early successes of some works on neural networks pointed to an overestimation of the artificial neural networks potential, also held back by the technological capacity of the time. Finally, important questions of philosophical nature came to light, such as the fear fueled debate on the impact on our society of a class of computers able to think. This very controversy, i.e. the artificial intelligence (AI) problem, is discussed still today [9].

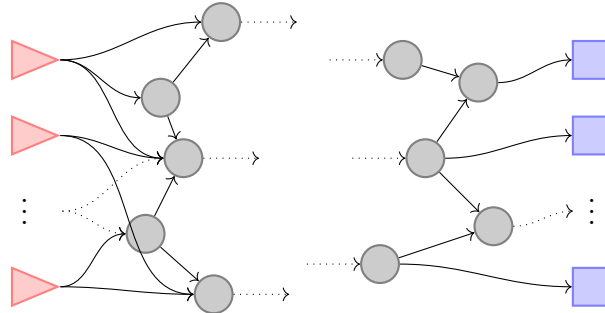
Sometime during the 1980s the interest for this computing method was reinvigorated. The main stimulation was probably given by a number of works which suggested methods to implement multi-layered networks distributing pattern recognition errors through the all the layers in the network. This method is now called *backpropagation*.

In today's research and technology, artificial neural networks are used in numerous applications. However, the development is made slowly, due to technological limitation in computational power of present processors.

## 1.0.2 Comparison with conventional computers

## 1.1 Basis of Neural Networks

A neural network is a collection of processing elements, or nodes, interconnected in an arbitrary topology. From its input nodes, the network accepts information, which will propagate into the inner nodes through the interconnections and will get elaborated at each node. At the end of the network, there will be a number of output nodes, with the task of reading a portion of the inner nodes. The inner nodes are also called hidden, because they are not meant to be accessible to the external world. A generic scheme of such network is shown in Figure 1.1 on this page.

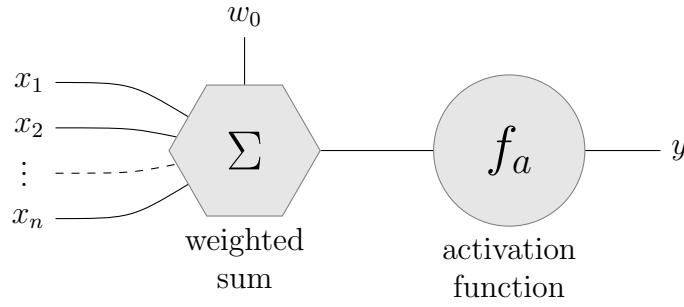


**Figure 1.1:** Generic scheme of a neural network. Triangles (red) are input nodes, circles (grey) are inner nodes, and squares (blue) are output nodes. Interconnections among nodes are represented by arrows: continuous when both elements are drawn, and dotted otherwise.

Nodes can all implement the same function or behave differently, depending on the type of neural network. The operation of nodes resembles that of animal neurons: various input gets collected and elaborated together to obtain an output, which will become one of the many inputs for subsequent neurons/nodes. Specifically, the most used model for neurons is the McCulloch–Pitts (MCP) neuron. It is divided into two parts, as shown in Figure 1.2: the first part is a weighted sum of the inputs, while the second part is given by the so called activation function.

The node is described mathematically by Equation 1.1

$$y = f_a \left( w_0 + \sum_{i=1}^n w_i x_i \right), \quad (1.1)$$

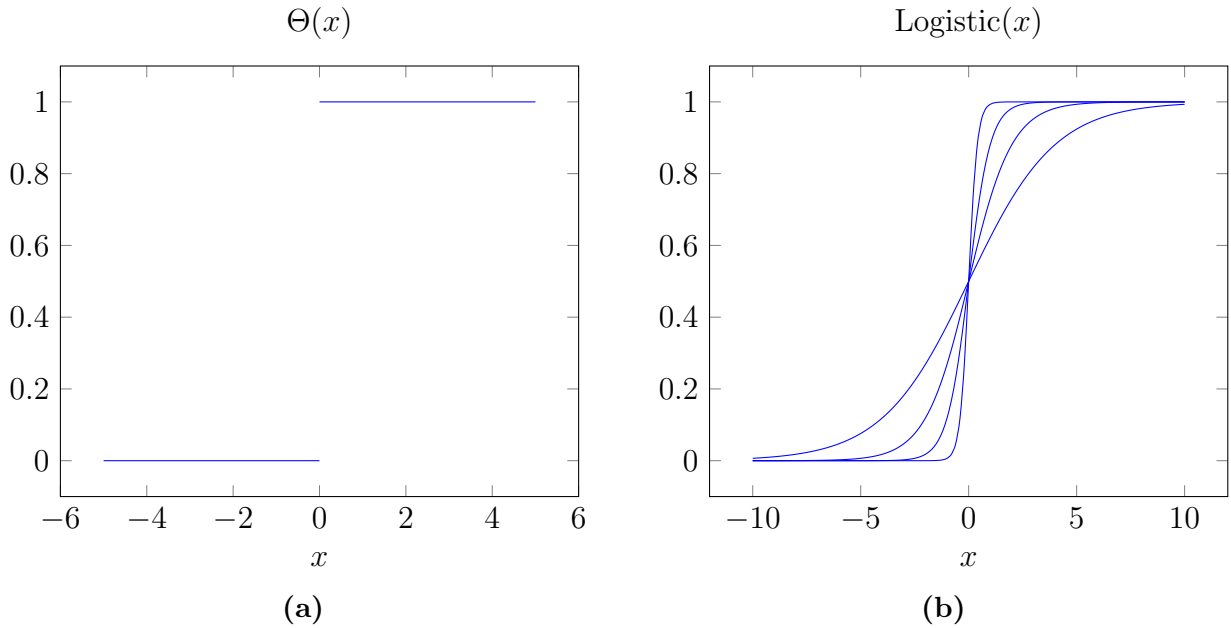


**Figure 1.2:** Generic node representation.  $x$ -values are inputs,  $y$ -values are outputs,  $w_0$  is the bias.

where  $f_a$  is the activations function, evaluated on the sum of the input  $x_i$  weighted with  $w_i$ , plus a bias  $w_0$ .

Each node accepts values at its inputs and produces an output accordingly. However, in addition to the input, the output depends also on the node's parameters: the weights and the bias, which are usually changed outside the operative phase of the neural network (see Section 1.2).

Moreover it is mandatory for the activation function  $f_a(\cdot)$  to be nonlinear, because otherwise a collection of nodes will result in just a weighted sum of its inputs. Two examples of nonlinear function are shown below in Figure 1.3.



**Figure 1.3:** Examples of activation function: (1.3a) is the well-known step function, or Heaviside  $\Theta$ , (1.3b) depicts a few functions from the family of the Logistic functions.

One can distinguish at least three type of nodes in every neural network: input, inner/hidden, and output nodes. Input nodes take one input value, from the outside of the neural network, and pass it on to the inner nodes unchanged. Inner/hidden nodes take many inputs and generate an output through the activation function. Output nodes, similarly to input nodes, take one input value, from the inside of the NN, and pass it on to the outside.

## Standard Representation

The way I depicted a generic neuromorphic network in Figure 1.1 is not the standard representation used in books and research papers. The main difference is that usually weights are commonly represented on the connection between the nodes, which are then designated to apply only the activation function. Moreover the input layer is linear as it feeds the inner nodes with the input data, while the output layer is actually given by the last nonlinear layer of the inner nodes. A generic network is shown in Figure 1.4a.

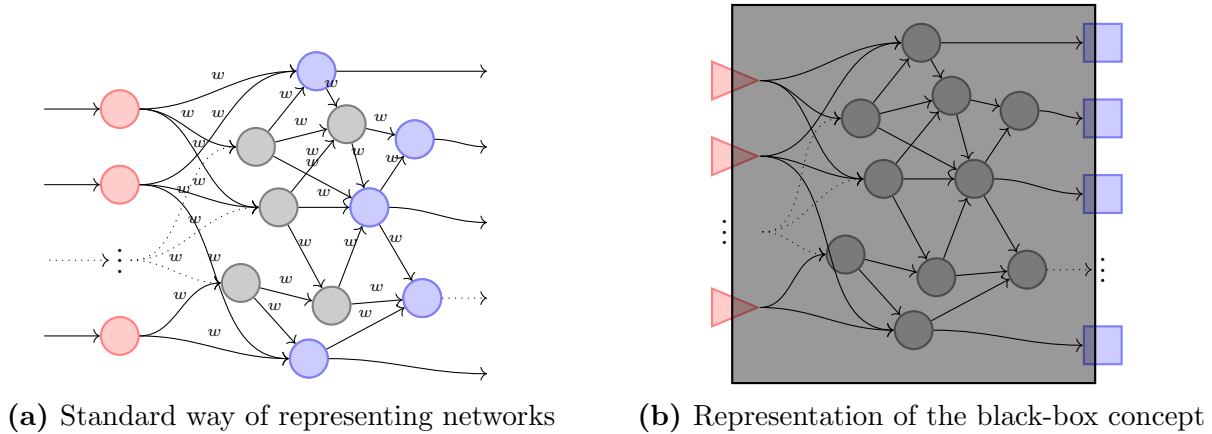


Figure 1.4

On the contrary, I consider the inner nodes as the only place where any kind of elaboration on the data happens. Inner nodes have a number of inputs, which are weighted and summed together to be entered as argument in the activation function. This leads to a natural separation between input/output nodes, which acquire the task of providing data from/to the outside, and inner nodes, which is where the activation function and/or the weighted sum are carried out.

This non-standard description, moreover, is consistent with the idea of functional *black box*, in which input and output are the only visible nodes, while the other are hidden inside, as shown in Figure 1.1.

## 1.2 Working Principles of ANNs

Because of its topology, each neural network will behave in a different manner from other neural networks with diverse, or even similar, arrangements of nodes. Moreover the same neural network will perform a certain task better or worse also depending on how inputs are weighted at each hidden node, and normally those parameters are initialized with a random value at the creation of the network. For this reason, before a neural network is considered ready to perform a task, it usually must go through three training stages: learning phase, validation phase, and testing phase. Every one of these stages is meant to prepare the network to work as required from the designer.

### 1.2.1 Learning Process

During the learning process the neural network is run on a set of known inputs  $x$ , each paired with its correct answer  $y$ , or target, in a second set of data. The neural network will produce at the output a third set  $\hat{y}$ , which should be as close as possible to the correct answers, when the network works properly. Typically weights among nodes are initialized randomly and the distance of the network outcome from the target function is measured through a loss function. Then weights are modified in order to decrease the loss function to its lowest possible value.

## Loss function

The loss function  $L(y, \hat{y})$  evaluates the difference between the predicted and the correct answer. Usually, this quantity is linked to the geometrical distance between the predicted output and the target  $|\hat{y} - y|$ .

The most common loss function is the *mean-square error*. Assuming to have an input set of  $N$  examples paired with the same number of targets, and that the outputs and the targets are composed by  $C$  values, or classes, the function becomes:

$$L(y, \hat{y}) = f_{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{n=1}^N \sum_{i=1}^C (\hat{y}_{n,i} - y_{n,i})^2, \quad (1.2)$$

where each example in the set is subtracted to its target and then squared. Finally the mean of all squares gives the expected result.

Another commonly used function is the cross-entropy loss (also known as negative log likelihood),

$$L(y, \hat{y}) = f_{CEL}(y, \hat{y}) = -\frac{1}{N} \sum_{n=1}^N \sum_{i=1}^C y_{n,i} \log(\hat{y}_{n,i}), \quad (1.3)$$

which expects positive values at the input. Hence the error  $-y \log(\hat{y})$ , quantified for each element in each example, is always a positive number. The mean over all examples in the set returns the results.

Alternatively, variations of the previous methods are given by taking the sum of the examples in place of the mean, or by calculating a loss function for each example instead of evaluating it for the whole set.

## Weights Update Process

The weights update process is a difficult task, and probably the most computationally expensive one in running a neural network. There is a variety of methods to chose from, depending on the type of artificial network and the resources available.

A widely used algorithm is the gradient descent, from its most simple version to more complex variations such as stochastic gradient descent (SGD). This method updates the weights by subtracting a value proportional to the gradient of the loss function in respect to the weights themselves times a positive factor called *learning rate*, as shown below.

$$w_i|_{n+1} = w_i|_n - lr \cdot \frac{\partial L}{\partial w_i|_n} \quad (1.4)$$

where  $w_i|_n$  are the current weights,  $lr$  is the learning rate,  $\frac{\partial L}{\partial w_i|_n}$  is the first derivative of the loss function in respect to the  $i$ -th weight at the current step, and  $w_i|_{n+1}$  are the updated weights. This method is equivalent to minimize the error on the loss function, by following the gradient  $\nabla_w L$ . This vector lives in the multidimensional space of the loss function  $L : \mathbb{R}^W \mapsto \mathbb{R}$ , where  $W$  is the total number of parameters in the network.

The most efficient ?? algorithm is called *backpropagation*: it computes the first derivative of the loss function  $L$  in respect to all the parameters of the network, the weights, starting from the end of the artificial network and going backward toward the input, hence the name backpropagation. Since the number of connections between nodes might be even order of magnitude bigger than the number of nodes, it is simple to understand how large networks are computationally expensive to train.

**Other types of learning processes** are used, e.g. unsupervised/reinforced.

### 1.2.2 Validation Process

### 1.2.3 Testing Process

At the end, there is the process of testing the artificial neural network. Ideally the network is tested on a new set of data, for which the target results are known, similarly to the preceding phases. This time, however, the predicted outputs are compared to the correct answers to obtain an overall value for the correctness, often expressed in percentage.

### 1.2.4 Datasets

Since there are three phases of preparation for any artificial neural network, there must be an appropriate number of examples to feed to it.

HOW DO I DIVIDE A DATASET?

WHAT HAPPENS WHEN THERE ARE TOO FEW EXAMPLES?

AND WHEN THERE ARE TOO MANY?

## 1.3 Feedforward NN

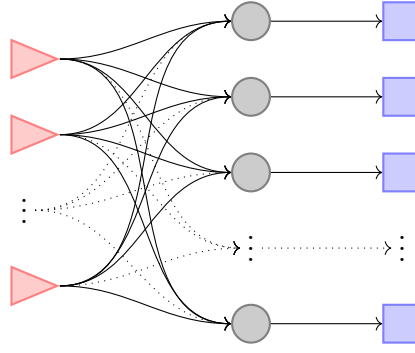
The first and most simple type of neural network is called Feedforward. In this kind of neural network, nodes are divided into groups called *layers*. A layer is a collection of nodes that accepts inputs from a preceding group and generate as many outputs as the number of nodes in the layer. Each layer of a Feedforward neural network is connected in series with the others, except of input layer at the beginning and the output layer at the end. As for the single nodes, the inner layer are called hidden, because usually not accessible.

The information travels from the input to the output and gets elaborated from each hidden layer: there are neither connection between nodes of the same layer, nor loops or feedback between layers. The number of hidden layers and the number of nodes they contain depends on the network topology. Moreover the connection between the layers might be complete, i.e. each node in the layer accepts each input of the preceding layer, in that case the layer is said to be *fully connected*, or sparse as in the case of convolutional layers (see Section 1.3).

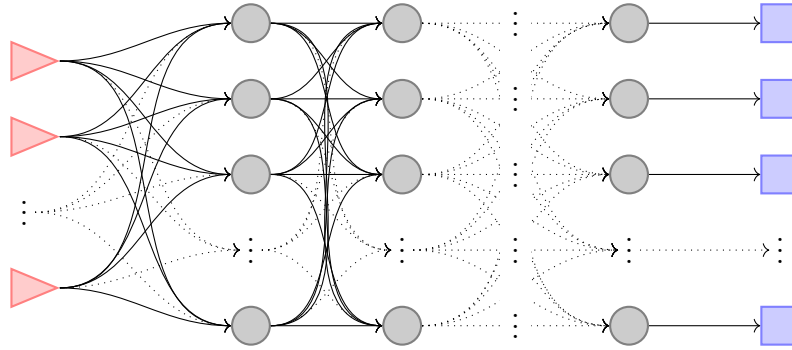
### Perceptron

The most naive topology of a Feedforward neural network is given by the so called *Perceptron*. The Perceptron dates back to the 1957, when the homonym *Perceptron algorithm* was software implemented by Frank Rosenblatt on a computer (IBM 704) and only subsequently in hardware as the *Mark 1 perceptron* [6], [7]. The graph of a generic (single layer) perceptron ANN is shown in Figure 1.5 below.

By adding more than one hidden perceptron layer to the neural network, one obtain the so called *Multi-Layer Perceptron* (MLP). This allows for more computational complexity ???. When the total number of layer is more than two, the network is called *deep*. A deep MLP is shown in Figure 1.6. In principle any shape is possible, i.e. each layer could have a different number of nodes, however often the layers at the beginning are wider than the layer at the end of the network ??. Besides the shape, in literature a perceptron is almost always considered fully connected ??.



**Figure 1.5:** Perceptron type neural network: in this representation the perceptron has  $n$  inputs and  $m$  outputs as well as a hidden layer with  $m$  nodes.



**Figure 1.6:** Deep Multi-Layer Perceptron (MLP), fully connected.

### Other Feedforward NNs

**Autoencoder** neural networks are feedforward networks in which the output nodes are as many as the input nodes. The purpose of this kind of network is to reconstruct its own inputs.

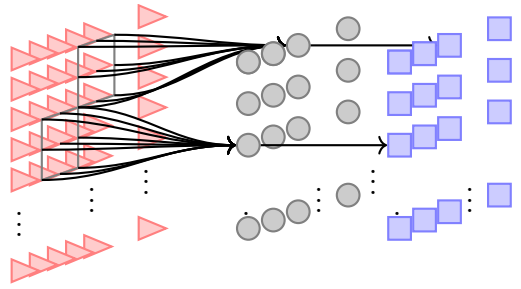
### Probabilistic

### Time delay

**Convolutional** networks are inspired to the visual cortex, in which neurons are not fully connected their inputs but only to a restricted region. Convolutional neural networks are a type of feedforward network conceived to recognize images without being misled by distortions such as translation, skewing, or scaling. In this kind of network the input is often represented by a 2D matrix, instead of a 1D vector. It is usually composed by many layers, the prevalent kind is the convolutional one. This layer performs a two-dimensional convolution over the input matrix of a second 2D matrix of weights, called *feature map*. Thus, each node of the layer operates on a restricted region to understand if a feature is present or not. Commonly the operating regions are overlapping and the feature map is shared among the nodes in the same layer.

#### 1.3.1 Other Types of NNs

By changing the topology of the nodes distribution and their connections, one obtain other networks that cannot be catalogued under the class of feedforward networks. Moreover, those

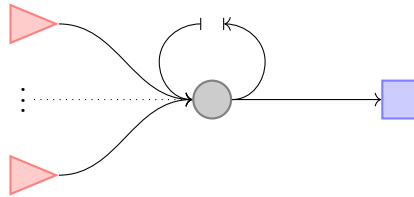


**Figure 1.7:** Pictorial representation of a layer of a convolutional supernode. Several supernodes form a convolutional layer.

different types of network are not a niche, but are widely ?? studied as a different approaches to the same or additional problems.

### Recurrent NN

Recurrent neural network are a kind of network in which a portion of the input of nodes depends on the (past) output of the same nodes or nodes of subsequent layers. That is information does not propagates only forward like in the feedforward networks, but can propagate also backward, for example in loops or in feedbacks.



**Figure 1.8:** Representation of a recurrent node. One of the inputs is given by the output itself. This output to input connection could be mediated by a delay device, so that for example the output at  $t - 1$  becomes the input at  $t$ . Depending on the structure of the network, there might be recurrent nodes and/or recurrent group of nodes, loops.

### Reservoir NN

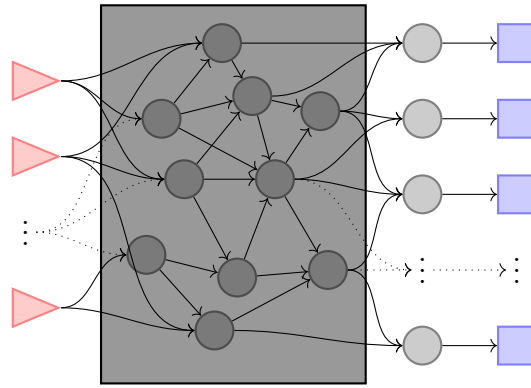
Reservoir neural networks differ from feedforward and recurrent networks in the learning approach. In fact, the topology of a reservoir network could be exactly the same as that of a deep multi-layer perceptron or that of a recurrent network. However, the reservoir computing differs in approach in respect to deep learning. It claims that it is not necessary to learn all the weights of the network, as in deep learning, but it is sufficient to train only the last (perceptron) layer of the network.

This kind of networks, then, can be trained much faster than their respective counterparts, i.e. feedforward and recurrent. The question over which training method is correct is still debated and literature does not provide clear answers yet.

### Modular NN

### Spiking NN

Spiking artificial networks are the most different kind in respect to all the other networks until now described. In this class of ANNs, information is not coded only in the intensity of the



**Figure 1.9:** A reservoir NN is topologically equivalent to deep networks. However only the last layer is trained. In this picture the trained node are represented outside the box, while those inside are initialized with random weights which are then left unchanged.

signal, but also in the rate of signals, e.g. a high value will be encoded as a signal with high repetition rate, whereas a low value as a signal with low repetition rate. This way of encoding information is more alike the mechanism of biological neural networks, such as our brain ??.

LOOK INTO other type of neuron model: Hodgkin–Huxley (H-H) model [https://en.wikipedia.org/wiki/Binding\\_neuron](https://en.wikipedia.org/wiki/Binding_neuron)

## 1.4 Real-Life Examples

? SHOULD I KEEP THIS SECTION ?

## 1.5 ANN Simulation

The resources and the time available allowed me to implement physically only the activation function of one node. Hence, to test this hardware implementation as I will show in Section 3.4 of Chapter 3, I had first to simulate and train *offline* a specific neural network. To do so, I chose a programming language, *Python*, and a library, *PyTorch*, which helped me in this task.

### 1.5.1 PyTorch

PyTorch is a python package that provides high-level features such as a deep learning research platform [10]. It is based on a backpropagation algorithm called *Reverse-mode auto-differentiation*, which is fast and flexible. Moreover integrates acceleration libraries that allow fast and lean operation.

To summarize, the library was chosen for its language and its flexibility, even though it is also powerful. This because our need was to simulate a little network to test the activation function.





# Chapter 2

## Integrated Photonics

To build an artificial neuromorphic network one has to choose first some physical phenomenon to employ in the fundamental blocks, likewise transistors in electronic circuits use the various behaviors of electrons in resistances, capacitors, and inductances. The physics which I want to build my artificial neuromorphic network with is photonics, precisely integrated photonics.

Photonics is the physical science which studies detection, manipulation, and emission of light. Specifically, integrated photonics is the branch that studies how to reduce and *integrate* once macroscopic optical devices in miniaturized structures. In the past few decades, driven by the growing needs of communication technology, which was in turn following the increase in computational power of electronics, many integrated devices have been proposed and commercialized.

### 2.0.1 Silicon Photonics ?

## 2.1 Guided-wave photonics

The most important thing for integrated photonics is the way light is confined and manipulated into microscopic structures. While in conventional optics, light is delivered with bulky mirrors and lenses toward the desired position, in integrated photonics waveguides are the main device for the task.

### 2.1.1 Waveguides

A waveguide is a path inside a medium, whose volume is defined by a certain number of interfaces between different materials, in which light remains confined and ideally travels with negligible losses. This volume is called *core* of the waveguide, while the medium external to it, if present, is called *cladding*.

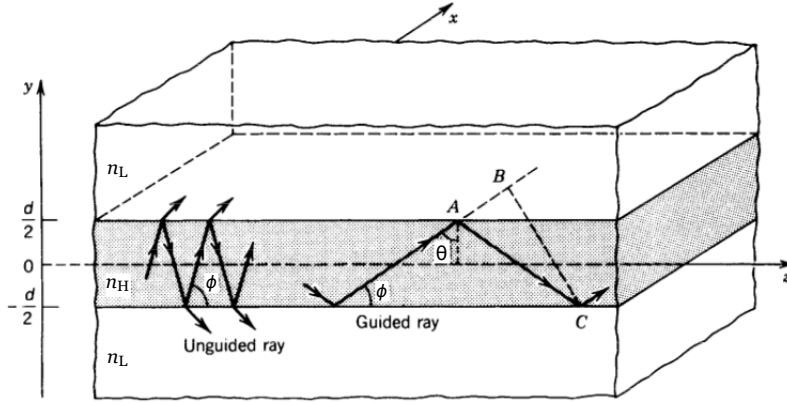
One can distinguish two main types of waveguides: metallic and dielectric. The former are based on the reflection of the electromagnetic field by the metallic surface. However, for visible and infrared frequencies, metallic waveguides are not possible, due to too high absorption of the metals at optical frequencies. On the other hand, dielectric waveguides are based on the phenomenon of total internal reflection (TIR) on the interface between two dielectric media. To produce TIR, the two materials must have sufficiently different real part of the refractive index. Moreover, they must also be transparent in the required range of frequencies. Hence not all media are suitable to build dielectric waveguides that work at a specific wavelength. For example, silicon is known to be transparent for light at and around  $1.55\text{ }\mu\text{m}$ , among other wavelengths. This feature together with a low distortion of signals is the reason why it is the most used wavelength in communication technology.

## Dielectric Waveguides

Total internal reflection is the well-known phenomenon in which light coming from a material with high refractive index  $n_H$  gets reflected at the interface with a material with low refractive index  $n_L$ . For this to happen, incident light must be at an angle greater than the critical angle given by the Snell's law:

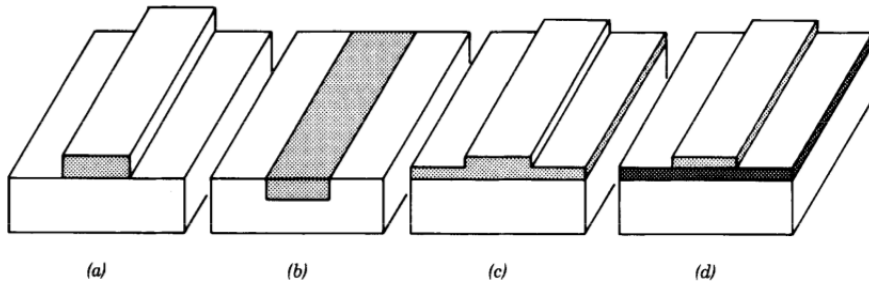
$$\theta_C = \arcsin \left( \frac{n_L}{n_H} \right).$$

Waveguides have many shapes, however one can initially discriminate them by the dimensionality of the confinement of light. The simplest one is called *slab* and it is composed by two interfaces, which divides the material in three volumes, as shown in Figure 2.1 below. This type is classified as a 1D-waveguide, because it constrains light in only one dimension.



**Figure 2.1:** Scheme of a dielectric slab waveguide, made by two materials with refractive indexes  $n_H$  and  $n_L$ . Two rays are shown: the unguided one is incident on the interface at an angle smaller than the critical angle. Conversely the guided ray is incident at an greater angle and is therefore totally reflected inside the waveguide [11].

2D-waveguides on the other hand are more common, to the point that with the term waveguide one usually refers to a device belonging to this category. Since they confine light in two dimension, their core then becomes a straight path with which one can conduct light from a place to another. A few types are shown in Figure 2.2 below.



**Figure 2.2:** Representation of a few types of 2D dielectric waveguides. Light is constrained in two direction and can only move forward or backward in the third direction. (a) strip, (b) embedded strip, (c) rib, (d) strip loaded [11].

Obviously, the versatility of straight (2D-) waveguides is limited. Hence more complex structures such as bent waveguides have been developed.

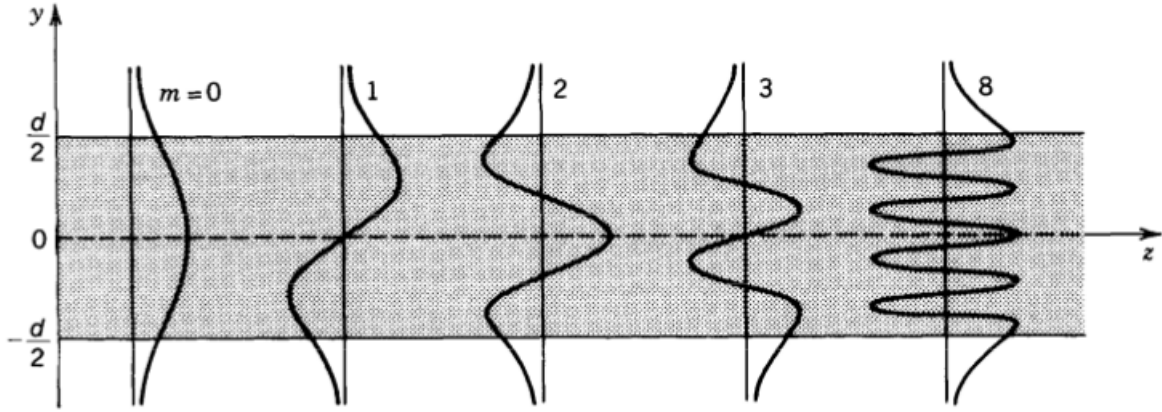
### Propagation of light inside waveguides

Light propagates inside waveguides as superposition of transverse electro-magnetic (TEM) waves which keep reflecting on the interfaces of the core. The result of the superposition is the so called *mode*, which is expressed mathematically by

$$\phi(\mathbf{r}) = \phi_m(x, y) e^{i(\beta_m z - \omega t)}. \quad (2.1)$$

$\phi_m(x, y)$  is the transverse field distribution,  $z$  is the direction of propagation,  $\beta_m$  is the propagation constant of the  $m$ -th mode, and  $\omega = 2\pi\nu$  is the angular frequency of light. Each mode, identified by the  $m$  index, travels inside the waveguide with a certain propagation constant  $\beta$  and maintaining a field distribution  $\phi_m$ . Both of them depends on the materials and the geometry of the waveguide, however while  $\beta_m$  decrease with increasing  $m$  indexes, the features of the function  $\phi_m$  grows in number. Light either propagate in a manner described by a mode or superposition of modes or is rapidly scattered away.

The field distribution of some of the first modes, in the simplest case of a slab waveguide, is represented in Figure 2.3 below. The function inside the core is characterized by maxima,



**Figure 2.3:** Field distribution inside a slab waveguide.

minima and nodes with a certain periodicity and the number of nodes (where the function is zero) is strictly linked to the index  $m$ . Outside, the distribution of the field decays exponentially. For waveguide with a 2D core, the field can be expected to be a multivariate version of the same or similar function. However, differently from the analytical solution for slab waveguides, they are almost always found numerically.

The propagation constant can be written as

$$\beta_m = n_{eff} k_0 \quad (2.2)$$

where  $k_0$  is the wavevector in vacuum and  $n_{eff}$  is the *effective refractive index* of the mode. The value of this quantity is between the core refractive index and the cladding refractive index  $n_L < n_{eff} < n_H$ . The physical interpretation of the effective index is that of an average refractive index *felt* by the mode propagating as a whole, instead of a superposition of multiple TEM waves *feeling* different refractive indexes depending on the specific medium in which they are propagating. Similarly to the actual refractive index of materials,  $n_{eff}$  is a complex number and the imaginary part is directly linked with the absorption properties of the materials. Actually, a low imaginary part means that the material absorb less light and is therefore transparent.

All of these parameters which describe the propagation of light along the waveguide depends both on the materials and the geometry of the core and the cladding, but also on the frequency of light. For dielectric waveguides, the fundamental mode ( $m = 0$ ) is always supported while

higher modes might not be, depending both on the waveguides and on the frequency of light. For this reason one distinguishes *single-mode* waveguides, which allow propagation for only the fundamental mode in a certain operative range of frequencies, from *multi-mode* waveguides, which allow higher order modes to propagate.

## mode propagation and GVD?

### 2.1.2 Light coupling

We covered how light travels inside a waveguide, now the problem on how to insert and extract light from it. The process of inserting light into a waveguide is called *coupling* and it often requires at the same time to extract light from another waveguide.

#### End coupling

The most naive way to insert light into a waveguide is obviously to use one of its ends. This is achieved either by radiating a beam directly on the whole waveguide or by focusing it inside the core with some lensing mechanism (e.g. tapered optical fibers).

These methods are very simple, however they have a low coupling efficiency.

#### Grating coupling

A more sophisticated way to couple light inside a waveguide is to use the so called *grating coupler*. A grating coupler is a periodic structure end coupled to a waveguide. It is designed in a way such that light incident from free space above it, at a certain angle and with a specific wavelength, is coupled inside the waveguide.

#### Evanescent coupling

Evanescent coupling exploits the fact that the field distribution outside the core decays exponentially to zero. This means that when two waveguides are separated by a sufficiently small distance, the field distribution of one waveguide cannot decay sufficiently fast to zero and instead extends itself over the core of the second waveguide. When this happens optical power can be transferred between the waveguides. Therefore, if at the beginning light is propagating inside one waveguide, light is *coupled* inside the other waveguide because its mode overlaps the core of the second waveguide. Depending on the parameters of the problems, light can couple completely or partially between the waveguides.

This exchange of optical power is periodic with their length.

### 2.1.3 Microring Optical Cavity

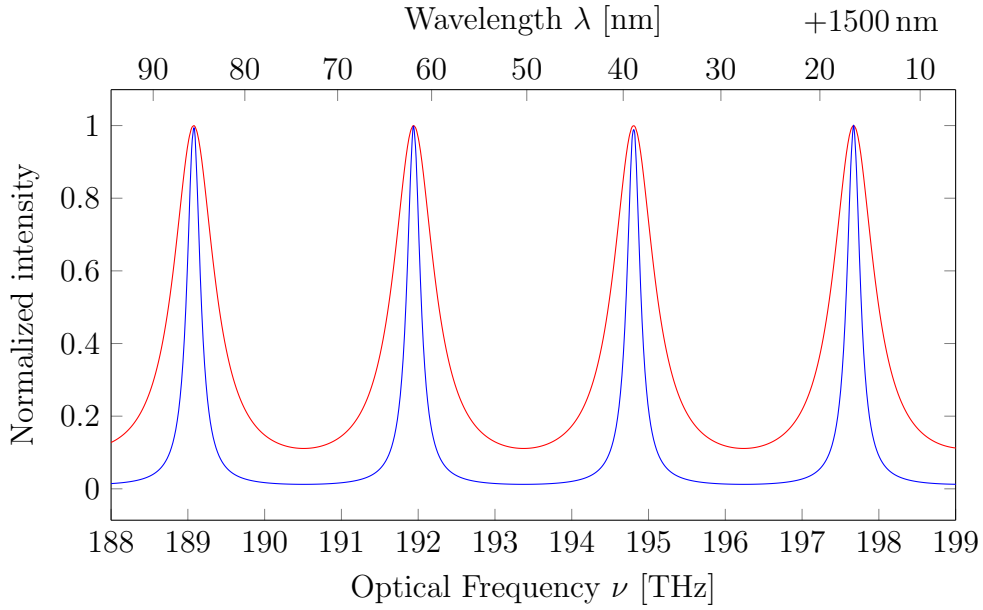
In integrated photonics the microring is an optical cavity made by bending a waveguide on itself. To insert light into the cavity, the microring is often side coupled with one or two straight waveguides. Light inserted from the different ports into the cavity interferes with itself, after a round trip, and with each other. When the waves are in phase, they generate constructive interference and the energy stored in the cavity increases exponentially. On the other hand, when the signals are out of phase, the interference is destructive and the energy stored in the cavity remains low.

In the simpler case, when there is only one waveguide side coupled to the resonator, the optical cavity has only two ports, which are usually called *input* and *through*. This configuration is called *All-Pass Filter*, because in the ideal case, where no losses happen, all the signal passes from the input to the through channels. All pass resonators have only one input and one output:

light enters the resonator from the input and exits from the output, unless the device is used backwards or some specific phenomena happens (e.g. back scattering).

### Add-drop filter theory

By adding a secondary couple of input and output ports, one obtains the so called add-drop resonator. The name comes from the fact that the secondary input is used as an add port to convey additional content to the output channel, while the drop port is used to take some of the light from the input channel. Of course, light might also travel from the input to the output and from the add to the drop. Whether light goes straight or passes through the resonator depends on the frequency of the signals and the relative position of the resonance. Resonators with more than four ports are possible, however are very uncommon.



**Figure 2.4:** Transmission spectra of microring resonators in Add-Drop Filter configuration for the through and drop ports.

#### 2.1.4 Nonlinear Perturbations

Comparison between nonlinear optics phenomena which affect a microring dynamic:

- TOC
- FCD
- FCA
- Kerr
- TPA

## 2.2 Integrated photonics applied to ANNs

Since our experiments are carried out in a time scale such that physics phenomena can be considered quasi-static, we obtain also that the only nonlinear effect non negligible is the thermo-optic effect.

### 2.2.1 Weighted sum of inputs

This has already been demonstrated and integrated widely, so it will not be the focus of this work. Two example are the banks of microring resonators and the MZ interferometers.

### 2.2.2 Nonlinear Activation Function

As opposed to the mechanism for weighted sum, an optical phenomenon for the activation function in an integrated photonic circuit has yet to be proposed.

### 2.2.3 Simulations

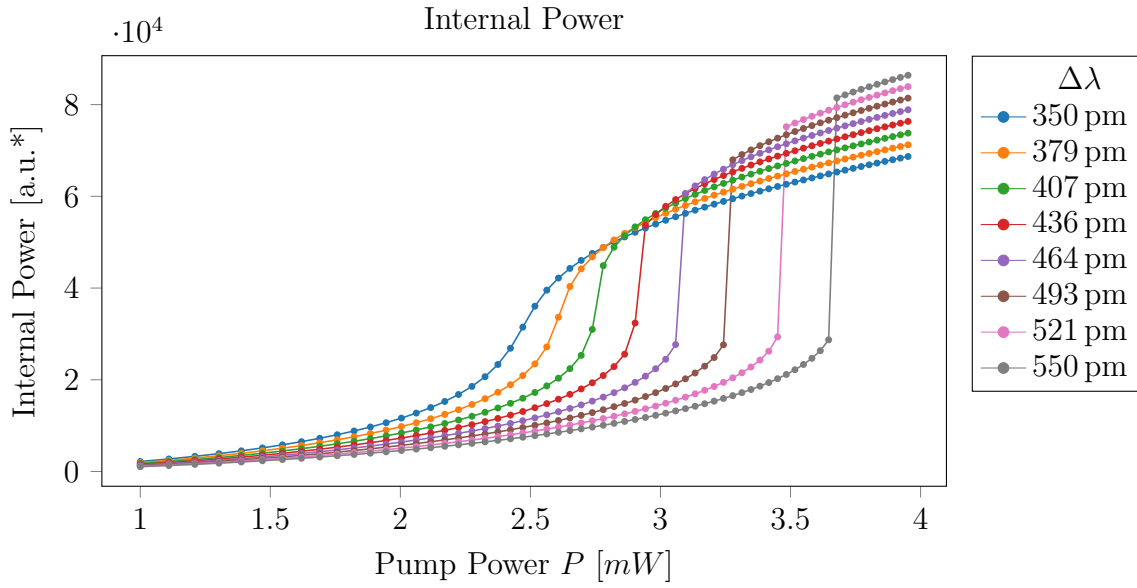


Figure 2.5

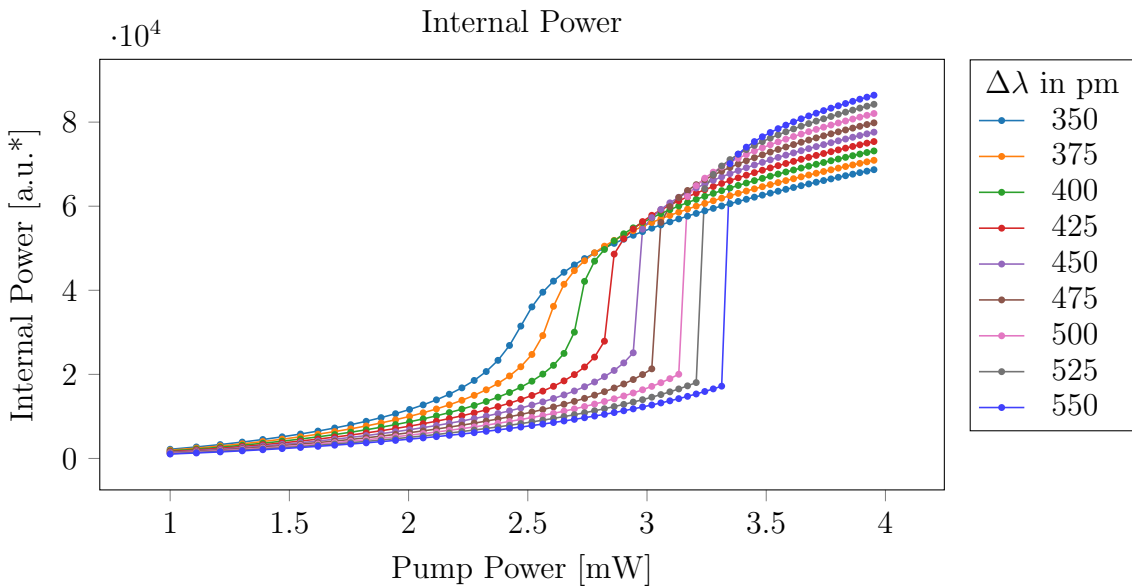


Figure 2.6

# Chapter 3

## Samples, setup and experiments

All the experiments have been made on samples manufactured for the IRIS project. This is due to the fact that in the time frame of this work there would have not been enough time to design and produce an ad hoc device. Moreover, as already stated, the aim of this thesis is to produce a proof of concept for an all-optical implementation of an activation function, rather than to construct a complete prototype.

### 3.1 The samples

The IRIS project studied the design and the implementation of an integrated reconfigurable silicon photonics switch matrix, a routing device, as a replacement for electronic devices used in the telecommunication industry. The completed integrated photonic circuit consists of a matrix of waveguides crossing each other and linked by couples of racetrack resonators, thermally-controlled. At the ends of the waveguides other structures, interleavers and AWGs, allowed many signals at different wavelengths to be multiplexed/demultiplexed on/from the same waveguide.

The complexity of such photonic circuit required the fabrication, for testing purposes, of each and every structure of which it is composed, with various production parameters. The collection of all these devices on a chip was produced in many samples. Isolated from the others, but also grouped together with few other elements. Specifically these test structures were disposed on a single chip and were accessible via grating couplers.

Since this work is at the beginning of the project, my choice was a system of intermediate complexity, so that it could be used both to test the activation function and the weighted sum.

The structure selected is the following: a simple waveguide, coupled to eight drop channels by a single or a couple of ring resonators each, nicknamed *mini-matrix*. In this family of devices, there were those built with single microrings, double microrings, single racetracks, or double racetracks. The final choice was to study the *mini-matrix* in which the coupling mechanism was provided by single ring resonators, because of its simpler transfer function in respect to double microrings or double racetracks.

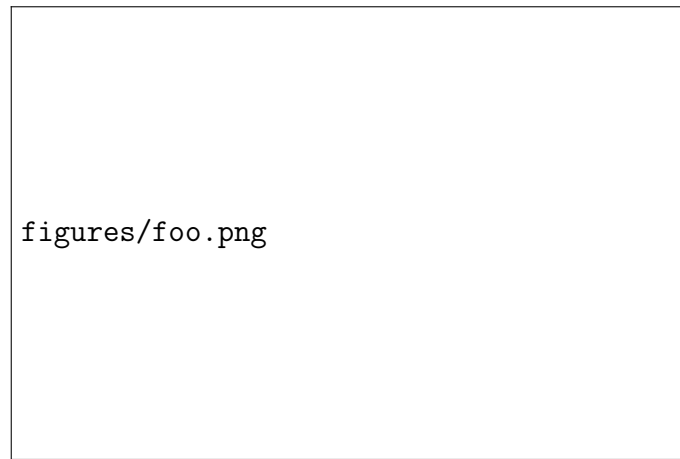
Moreover, these samples were manufactured with thermo-electric ?? pads to heat the rings and effectively tune their resonance.

### 3.2 Setup

### 3.3 Characterization of the Activation Function

Characterization of the activation function





**Figure 3.1:** image/scheme of the minimatrix

## 3.4 Test of a Trained ANN

Test of the activation function

# Conclusion

Furthermore, regarding the second objective, my work is part of a bigger project, BACUKP, with a wider scope and a more ambitious goal. It attempts to connect three fields of research, normally quite distant from each other: physics, computing technology, and biology. The main idea is to use the integrated photonics framework (physics) to build an artificial neural network (computing technology) on chip, where one could grow and, at least partially control, real neurons (biology). The aim is to develop a methodologies and devices that will allow us to study neural activity from a bottom-up perspective.



# Bibliography

- [1] *Humanbrainproject.eu*. [Online]. Available: <https://www.humanbrainproject.eu/en/silicon-brains/>.
- [2] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943, ISSN: 00074985. DOI: [10.1007/BF02478259](https://doi.org/10.1007/BF02478259). arXiv: [arXiv:1011.1669v3](https://arxiv.org/abs/1011.1669v3).
- [3] D. O. Hebb *et al.*, *The organization of behavior: A neuropsychological theory*, 1949.
- [4] B. Farley and W. Clark, “Simulation of self-organizing systems by digital computer”, *IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 76–84, 1954, ISSN: 2168-2690. DOI: [10.1109/TIT.1954.1057468](https://doi.org/10.1109/TIT.1954.1057468). [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1057468>.
- [5] N. Rochester, J. H. Holland, L. H. Haibt, and W. L. Duda, “Tests on a cell assembly theory of the action of the brain, using a large digital computer”, *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 80–93, 1956, ISSN: 21682712. DOI: [10.1109/TIT.1956.1056810](https://doi.org/10.1109/TIT.1956.1056810).
- [6] F. Rosenblatt, “The perceptron a perceiving and recognizing automaton”, *tech. rep., Technical Report 85-460-1*, 1957.
- [7] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain”, *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, ISSN: 0033295X. DOI: [10.1037/h0042519](https://doi.org/10.1037/h0042519). arXiv: [arXiv:1112.6209](https://arxiv.org/abs/1112.6209). [Online]. Available: <http://psycnet.apa.org/journals/rev/65/6/386.pdf%7B%5C%%7D5Cnpapers://c53d1644-cd41-40df-912d-ee195b4a4c2b/Paper/p15420>.
- [8] M. Minsky and S. Papert, “Perceptrons: An Introduction to Computational Geometry”, *MIT Press*, p. 268, 1969, ISSN: 00189340. DOI: [10.1109/T-C.1969.222718](https://doi.org/10.1109/T-C.1969.222718).
- [9] C. Clabaugh, D. Myszewski, and J. Pang, *Neural networks - history*, 2000. [Online]. Available: <https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/index.html>.
- [10] *Pytorch.org*. [Online]. Available: <http://pytorch.org/about/>.
- [11] B. E. a. Saleh, M. C. Teich, S. Editor, and J. W. Goodman, *Fundamentals of Photonics (Wiley Series in Pure and Applied Optics)*. 1991, vol. 5, p. 992, ISBN: 0471839655. DOI: [10.1002/0471213748.ch1](https://doi.org/10.1002/0471213748.ch1).



# Index

critical angle, 16

Snell's law, 16



# Acknowledgements