Università degli Studi di Padova

Department of Information Engineering

Master Thesis in Computer Engineering

Machine Learning based Signal Generation Strategies for High Speed Optical Transmitters

Supervisor
Prof. Fabio Vandin
Università degli Studi di Padova

Co-supervisor
Dr. Cleitus Antony, Prof. Paul Townsend
Tyndall National Institute, University College Cork

Master Candidate
Andrea Rossi

14th October 2019
Academic Year 2018/2019

To my father

Abstract

Optical communication is the only viable solution to respond to the demand for a high bit rate and long transmission distance. Directly modulated lasers (DMLs) are a cheap solution for modulating the light in optical fibre. Moreover, they consume less power and their hardware is simpler than externally modulated lasers. However, DML is inherently chirped and, due to chromatic dispersion in optical fibre, the transmission length with high bit rate is limited.

This work explores and implements neural networks based signal pre-distortion schemes to create optimum transmitter drive waveforms to counteract nonlinear distortions in the overall system. In the case of a DML system, the pre-distorted signal that precompensates the system impairments is unknown. For this reason, the training of the network cannot be done conventionally and requires indirect strategies.

The project demonstrates that these schemes are promising and opens to the possibility of reducing the complexity of the receiver, by using a digital signal processing (DSP) block at the transmitter, with the opportunity of increasing the bandwidth-length (BL) product without using expensive hardware.

Contents

Abstract v

List of figures ix

List of tables xi

List of acronyms xiii

1 Introduction 1
- 1.1 Motivation and Objectives of this Work . . . 1
- 1.2 Outline of the Thesis . . . 2

2 Optical Communication and Directly Modulated Laser 5
- 2.1 Optical Communication . . . 5
  - 2.1.1 Historical Perspective . . . 7
- 2.2 Fibre-optic Communication Systems . . . 9
- 2.3 Evaluation of System Performance . . . 12
  - 2.3.1 Bit Error Rate . . . 12
  - 2.3.2 Eye diagram . . . 15
- 2.4 Laser Diodes and Directly Modulated Laser . . . 16
  - 2.4.1 LASER . . . 16
  - 2.4.2 Directly Modulated Laser . . . 18

3 Artificial Neural Networks 27
- 3.1 Feedforward Neural Networks . . . 27
- 3.2 Neural Network Training . . . 31
  - 3.2.1 Cost function . . . 31
  - 3.2.2 Stochastic Gradient Descent and Backpropagation . . . 33
  - 3.2.3 Gradient Descent variants . . . 34
  - 3.2.4 Overfitting and Regularization . . . 39
  - 3.2.5 Hyperparameter Tuning and Validation . . . 41
- 3.3 Neural Network Application: Transmission System Regression . . . 42
  - 3.3.1 Dataset . . . 44
  - 3.3.2 NN Architecture and Training . . . 44
  - 3.3.3 Results . . . 45

3.4 Convolutional Neural Networks . . . 47
3.5 CNN Application: Eye Diagram Analyzer . . . 50
3.5.1 Dataset . . . 50
3.5.2 CNN Architecture and Training . . . 51
3.5.3 Results . . . 53

4 Particle Swarm Optimization 55
4.1 PSO Algorithm . . . 55
4.1.1 Hyperparameters and Update Rule . . . 57
4.1.2 PSO Applied to Neural Networks . . . 60
4.2 Comparison of PSO and SGD . . . 60
4.2.1 Equalization . . . 61
4.2.2 Dataset and Training . . . 62
4.2.3 Results . . . 63

5 Predistortion Techniques 67
5.1 Indirect Learning . . . 67
5.1.1 Motivation . . . 67
5.1.2 Implementation Details . . . 69
5.1.3 Results . . . 69
5.2 PSO for Predistortion . . . 72
5.2.1 General Overview . . . 72
5.2.2 Implementation Details . . . 73
5.2.3 Results . . . 75

6 Conclusion and Future Work 81

References 83

Acknowledgments 89

Listing of figures

1.1 Transmission technologies distance versus data rate 2

2.1 Optical fibre model and optical fibre end-face 6
2.2 Bandwidth-distance product evolution 7
2.3 FTTH network 9
2.4 Optical communication system model 9
2.5 Attenuation curve and transmission windows 11
2.6 BER curve 13
2.7 Bit error probability without ISI 14
2.8 Bit error probability with ISI 15
2.9 Eye diagrams 16
2.10 Eye diagram measurements 17
2.11 Laser spectrum 18
2.12 Optical spectrum of DML 19
2.13 Directly Modulated Laser operational characteristic 20
2.14 Direct modulation laser curve 21
2.15 Chromatic dispersion 22
2.16 Simulated chirp 24
2.17 Chirp 24
2.18 Effect of transient chirp 25

3.1 Activation functions for neural networks 28
3.2 Architecture of a 2-layers neural network. 29
3.3 Forward propagation 30
3.4 Example of gradient descent iteration. 34
3.5 Comparison between gradient descent and stochastic gradient descent . . . 35
3.6 Comparison between gradient descent and momentum. 37
3.7 Comparison between gradient descent and RMSProp. 38
3.8 Model complexity trade-off 40
3.9 Dropout in neural networks 42
3.10 Experimental setup for regression 43
3.11 Dataset input of NN application 44
3.12 Histogram of the actual transmission 46
3.13 Histogram of the emulated transmission 46
3.14 Prediction results of NN application 47

3.15 An example of 2-D convolution 48
3.16 Max pooling and average pooling 49
3.17 An example of CNN architecture 50
3.18 Eye diagrams dataset example 51
3.19 Training and validation loss in CNN application 53

4.1 Particle swarm optimization evolution 56
4.2 Schematic particle movement 58
4.3 Equalization model 61
4.4 Eye diagram and histogram after 100 km without equalization 64
4.5 Performance comparison for equalization 65

5.1 Indirect learning architecture 68
5.2 Eye diagram and histogram without optimization 70
5.3 Eye diagrams evolution using ILA 71
5.4 Histograms evolution using ILA 72
5.5 Schematic of MATLAB simulation setup for PSO 73
5.6 Example of signal generation using a LUT 74
5.7 Sigmoid function and modified sigmoid function 74
5.8 Waveform and chirp before and after optimization 76
5.9 Eye diagram before and after optimization 77
5.10 Stochastic BER before and after optimization 78
5.11 Comparison of `fminsearch` function and PSO perfomance 79

Listing of tables

2.1 Transmission windows ranges of the optical fibre. 11
2.2 Further subdivision of the optical transmission windows. 11

3.1 Summary of NN architectures 45
3.2 Summary of CNN architecture for 28×28 images 52
3.3 Summary of CNN architecture for 64×64 images 52
3.4 Results of the training and accuracy on the training, validation and test sets in CNN application 54

4.1 NN - Adam parameters 63
4.2 NN - PSO parameters 63

List of acronyms

ADC ..... Analog-to-digital Converter

AM ..... Amplitude Modulation

AWG ..... Arbitrary Waveform Generator

BER ..... Bit Error Rate

BL ..... Bandwidth-Length

CML ..... Chirp Managed Laser

CNN ..... Convolutional Neural Network

CD ..... Chromatic Dispersion

DAC ..... Digital-to-analog Converter

DFE ..... Decision Feedback Equalizer

DML ..... Directly Modulated Laser

DSP ..... Digital Signal Processing

EDFA ..... Erbium-Doped Fiber Amplifier

ER ..... Extinction Ratio

FIR ..... Finite Impulse Response

FFE ..... Feed-forward Equalizer

FM ..... Frequency Modulation

FTTH ..... Fibre To The Home

GD ..... Gradient Descent

ILA ..... Indirect Learning Architecture

IoT ..... Internet of Things

ISI ..... Intersymbol Interference

LASER ..... Light Amplification by Stimulated Emission of Radiation

LED ..... Light-emitting Diode

LUT ..... Look-up Table

MLP ..... Multilayer Perceptron

MSE ..... Mean Squared Error

MZM ..... Mach-Zehnder Modulator

NN ..... Neural Network

NRZ ..... Non-return-to-zero

OSA ..... Optical Spectrum Analyzer

PRBS ..... Pseudorandom Binary Sequence

PSO ..... Particle Swarm Optimization

PM ..... Phase Modulation

ReLU ..... Rectified Linear Unit

SDM ..... Space-division Multiplexing

SFM ..... Single-mode Fibre

SGD ..... Stochastic Gradient Descent

TDM ..... Time-division Multiplexing

WDM ..... Wavelength-division Multiplexing

1 Introduction

The growth in bandwidth demand to access the internet is continuously incresing over the years due to numerous services avaialable across the globe like video streaming, internet television and cloud storage. The development of internet of things (IoT) devices, included connected vehicles, home automation and wearable technology, leads the growth to increase more and more. Communication technologies are summarised in Figure 1.1 where they are compared in term of data rate and transmission distance [1]. It is clear than the viable solution which allows us to achieve high bitrate and long transmission distance is the use optical communication technology.

However, to respond to the high demand of bandwidth, there are some problematics to deal with while transmitting over optical fibre, due to the non-linearities of the channel. Moreover, there is interest in finding a cheap solution, applicable to optical acess (range 20-100 km) close to users.

1.1 Motivation and Objectives of this Work

Directly modulated lasers (DML), as one of the simplest light modulation form, have the advantage of lower costs and lower power consumption compared to externally modulated lasers. However, at high bitrates, the non-linearities in the optical fibre due to chromatic dispersion combined with direct detection limits the maximum allowable transmission length because pulse spreading due to dispersion degrades the quality of the received signal. This

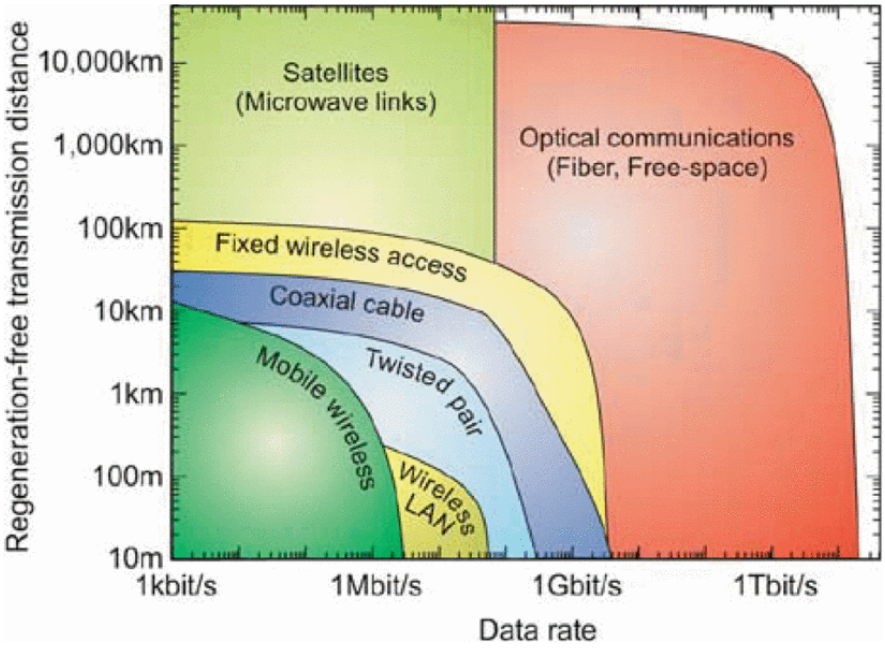


Figure 1.1: Transmission technologies distance versus data rate in regeneration-free transmission [1]

effect is made worse when the launch signal is chirped like in the case of DML.

Techniques like equalization at receiver have been proved to be effective while dealing with the problematics of DML, by using either linear filters or neural networks (NNs). However, there is interest in developing, producing and marketing light receivers, in which the digital signal processing (DSP) block is moved to the transmitter side. For this reason, a DSP block must be able to predistort the transmitted signal in order to compensate the non-linearities of the modulator and the channel, with the additional problem given by the fact that the target is unknown.

In this work, two predistortion schemes are proposed to overcome the limitations of a directly modulated laser system. In particular, these schemes, whose aim is to push the bandwidth-length (BL) product as much as possible, exploit NNs and their ability to represent a wide variety of non-linear functions, as universal approximators.

1.2 Outline of the Thesis

We will focus this work on developing predistortion schemes using neural networks to deal with the problematics of directly modulated laser. The general context and the motivation for this research have been introduced in Chapter 1. In Chapter 2 a summary of the optical

communication features is presented, focusing on directly modulated laser and the challenging aspects to deal with for increasing the performances. In Chapter 3 an overview of the neural networks and convolutional neural networks (CNNs) is presented; afterwards the attention will be moved on the use of neural networks as an universal approximator and the use of convolutional neural networks in the image processing context. In Chapter 4 an introduction to the particle swarm optimization (PSO) algorithm is presented; afterwards the performance of the algorithm is compared to the traditional methods used for equalization at the receiver. In Chapter 5 two predistortion schemes using indirect learning architecture (ILA) and particle swarm optimization are presented with the results of their application on a simulation testbed.

2

Optical Communication and Directly Modulated Laser

Optical communication is a branch of telecommunication in which light is used to transmit information at distance. Due to increasing of demand for high bit rate and transmission distance, optical fibre is the primary channel used for transmission. However, depending on the technology used to emit the light and the way it propagates along the fibre, there are serious challenges to overcome in order to achieve good performance in terms of speed and reliability.

2.1 Optical Communication

The basic optical fibre communication system includes a data source, an optical transmitter, an optical channel and an optical receiver. The data source consists of any signal sources, typically coded in binary formats, like images, audio, text, and transformed into an electrical signal. The optical transmitter and modulator are used to convert the data source signal into an optical signal. The channels are the carrier of the signal from the source to the destination. The optical receiver detects the optical signal and extracts the information from it, by converting it into electrical signals, in order to recover the originally transmitted message.

Nowadays, optical communication is widely used, the majority of which exploits fibre-optic cables as a transmission medium. An optical fibre is a thin, flexible, transparent fibre

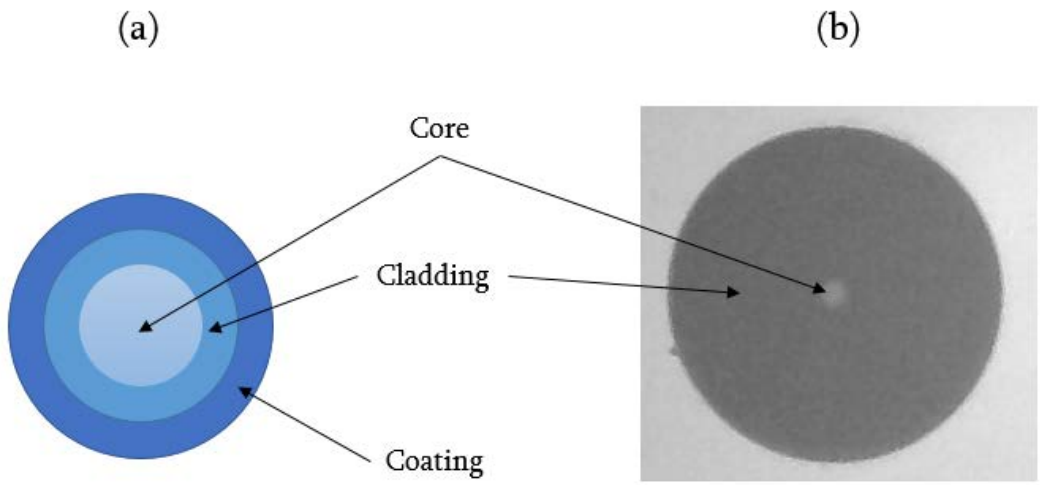


Figure 2.1: An optical fibre consists of a core, cladding, and coating. The figure shows (**a**) a simplified model of an optical fibre cable and (**b**) the end-face of a fibre captured using a fiber optic inspection microscope.

made of dielectric material like glass or plastic. Optical fibre is typically composed of a core, surrounded by a cladding characterized by a lower index of refraction, allowing the light to be kept in the core by means of total internal reflection [2]. The fibre is protected from the physical environment by one or more coats of a plastic material called outer coating (see Figure 2.1 (a) for the model and Figure 2.1 (b) for the end-face of a fibre captured using a fiber optic inspection microscope).

Since the telephone's advent, copper cables have been the dominant technology for wired data transmission and they were designed especially for voice calls. For current applications, fibre is becoming the dominant technology and copper cables are not suitable anymore, due to their high attenuation, that hinders transmission over long distances, and their limited bandwidth capability, through which it is not possible to reach high bitrates. The advantages and disadvantages of optical fibres are summarized below [3].

- Safety: Optical fibre does not transmit electricity and cannot be tapped, while its counterpart does, which can cause the entire system to fail. Moreover, there are different monitoring techniques that can be used to quickly detect breaks or damages that are not applicable to copper cable.
- Weight: Optical fibre is much lighter than any electrical cable.
- RF effects: Fibre is immune to several environmental factors such as temperature or electromagnetic interference, and therefore, it is more reliable than copper cables.

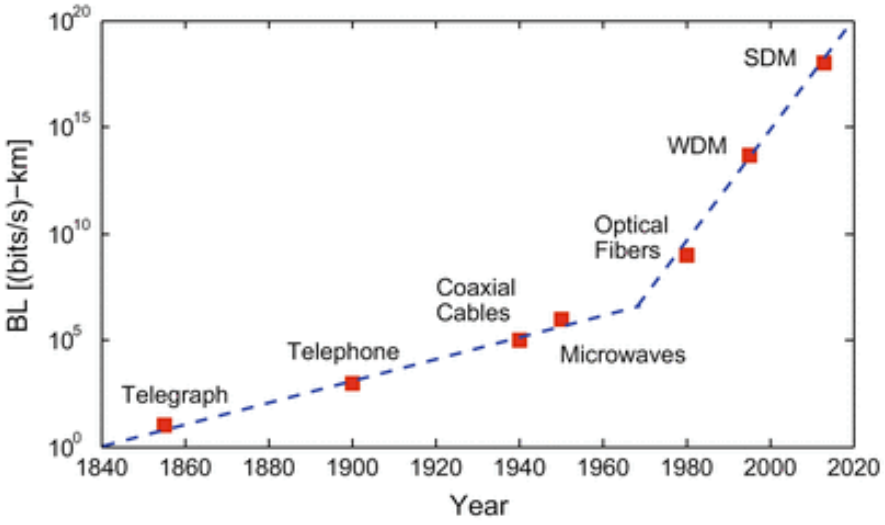


Figure 2.2: Increase in the BL product during the period 1840–2015 [4].

- Data bandwidth: Copper cable has limited bandwidth that means very high losses at high frequencies, with a 100-meter limitation for unshielded twisted pair copper, while fibre bandwidth is over $1,000$ times as much bandwidth as copper and can travel more than 100 times further.
- Durability: Both fibre and copper are reasonable in terms of durability.
- Cost: A few years ago, the overall price of optical fibre was more than twice that of copper, but today the price of optical fibre has decreased. However, the dominant cost is due to the transmission system used in optical communication.

2.1.1 Historical Perspective

A fibre-optic communication system is often characterized by its bandwidth-distance product BL [MHz · km], where B is the bit rate and L is the length after which the signal must be regenerated to maintain its reliability. For example, a fibre with a bandwidth-distance product of 1 GHz · km could carry 1 GHz signal for 1 km or a 500 MHz for 2 km. With the advent of optical fibre, the BL started growing exponentially, signalling the beginning of a new optical communication era. The growth of the BL over the time is shown in Figure 2.2. The current challenge is to increase the BL product to meet growing consumer demands.

Optical fibres were available during the 1960s and they were used only for short length applications because of impurities of glass that caused high losses after only a few meters [4]. At

that stage, no one was thinking about using optical fibre for telecommunication applications. In the following years, engineers discovered how to drastically reduce the impurities on the silica glass and the idea of using fibre in optical communication was considered revolutionary because it was able to guide light in a similar way copper wires guide electrons.

The evolution of the optical fibre technology from around 1975 through to today can be divided into different generations, each of which is characterized by the introduction of a new technology up to its saturation when it becomes matured. The progress of research on optical fibre and its applications has been distinguished by a succession of new technologies that improved the capacity, maximum transmission length and the safety of the cables.

The first fibre available commercially in 1980 was characterized by a bit rate of 45 Mbit/s and repeaters after 10 km and it was able to carry up to 700 calls at a time through the use of time-division multiplexing (TDM) [5].

By 1990, the bit rate started rising up to 2.5 Gbit/s with the use of dispersion-shifted fibres [4] and at the end of 1995 was 80 Gbit/s by exploiting wavelength division multiplexing (WDM) technique [6]. Recent progress has been achieved by using the modulation format with a better bandiwidth spectrum efficiency.

A type of fibre, called *single-mode fibres* (SMF) is used for longer distances and it supports only a single propagation mode. The core is smaller and requires more expensive components and interconnection methods. The use of SMF is allowing researchers to set new records up to tens of Tbit/s for hundreds of kilometres lengths.

The advent of the Internet in the early 1990s made it necessary to develop a worldwide network which allowed computers to be connected through the use of submarine cables across oceans. After 2010 a new technique called space-division multiplexing (SDM) [7] has been introduced, in order to respond to the increase of data traffic after the introduction of video streaming services by companies such as YouTube and Netflix.

Fibre cables have drastically revolutionized the world and they have been applied to many other industries, also from non-intrusive surgical methods (endoscopy) to military and space applications, from the automotive industry in safety applications like traction control and airbags, to lighting for house, streets and offices.

However, its main application is in the telecommunication field. In computer networks, optical fibre links allow faster transmission of data, while with the advent of high definition televisions, optical fibre cables are widely applied to cable television due to their greater bandwidth because of the high demand for high bandwidth services and reduced costs of the production of fibre cables. Because the bottleneck of the network is the last-mile, the

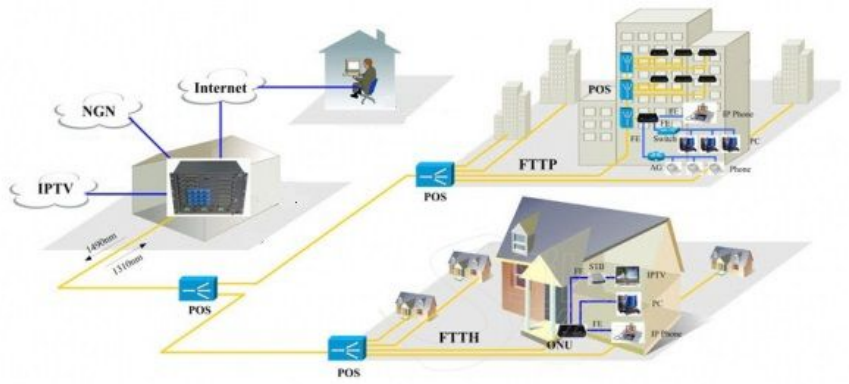


Figure 2.3: An example of FFTH network

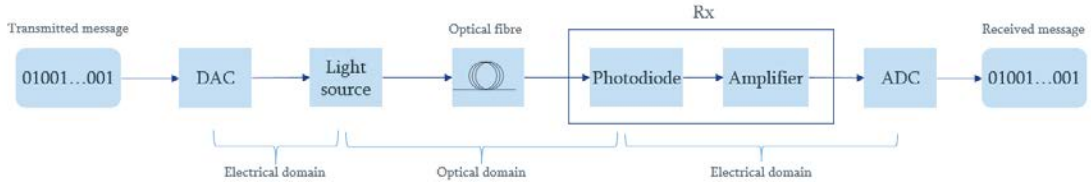


Figure 2.4: An optical communication system is composed of a transmitter, a channel and a receiver.

recent of fibre to the home (FTTH) [8] is replacing the old connections made by twisted cables, improving the performance of the network and the user experience. An FTTH network makes use of many types of fibre cable assemblies to bring connectivity to single homes, multi-dwelling units, apartment buildings, office buildings, light commercial buildings, etc. (Figure 2.3).

In order to improve the performance of the transmission, that is increase the BL product, the whole communication system must be characterized. Below, some tools for the diagnostics and the evaluation of the performance are introduced.

2.2 Fibre-optic Communication Systems

As mentioned in the previous sections, an optical communication system is composed of: a transmitter which encodes information onto an optical signal; a channel which carries the message to its destination; and a receiver which converts light into electricity in order to recover the transmitted message. The block diagram for an optical communication system is depicted in Figure 2.4.

The message, corresponding to a sequence of bits, is converted to an electrical signal by

means of a digital-to-analog converter (DAC) with a certain resolution, according to the level of quantization. The output of the DAC is converted into an optical signal through a light source and modulator, usually a laser diode or a light-emitting diode (LED). In this dissertation, the light source used in the simulations and experiments is a laser diode, whose operation will be illustrated in the next sections. The optical message travels through the optical fibre until the receiver, composed of a photodiode which converts the signal from the optical domain to the electrical one. Finally, an analog-to-digital converter (ADC) is used in order to recover the initially transmitted sequence of bits.

The transmission is performed by modulating a carrier waveform with a modulating signal (the message to be transmitted). An optical waveform as a function of time can be expressed as

$$\psi(t) = A\sin(2\pi ft + \varphi) = A\sin(\omega t + \varphi) \tag{2.1}$$

where A is the amplitude, f is the frequency, ω is the angular frequency and φ is the phase. This waveform can be modulated by varying the amplitude (amplitude modulation, AM) that is the amplitude of the carrier signal varies with the amplitude of the modulating signal, by varying the frequency (frequency modulation, FM) that is the frequency of the carrier signal varies with the amplitude of the modulating signal, or by varying the phase (phase modulation, PM), where the phase shift of the carrier signal varies with the amplitude of the modulating signal.

Usually, the transmission band is identified by the wavelength λ rather than the frequency f. Recalling that the relation $\lambda f = v$ holds for electromagnetic wave propagation, where v is the speed of the light in the medium, which can be expressed as $v = c/n$ with $c = 3 \cdot 10^8$ m/s the speed of light in the vacuum and n the refractive index of the medium.

The signal transmitted through the optical fibre is subjected to attenuation, whose behaviour, as a function of the wavelength, is shown in Figure 2.5. The useful interval for transmission is in the range from 800 to 1600 nm and there are three windows in which the attenuation shows a minimum, whose range is listed in Table 2.1.

The first window is suitable only for short-distance transmission, while despite the fact the second window has a loss of silica fibres much lower, the fibre amplifiers are not as good as their third window counterparts.

The range between the beginning of the second window and the end of the third one is further subdivided (Table 2.2) [10].

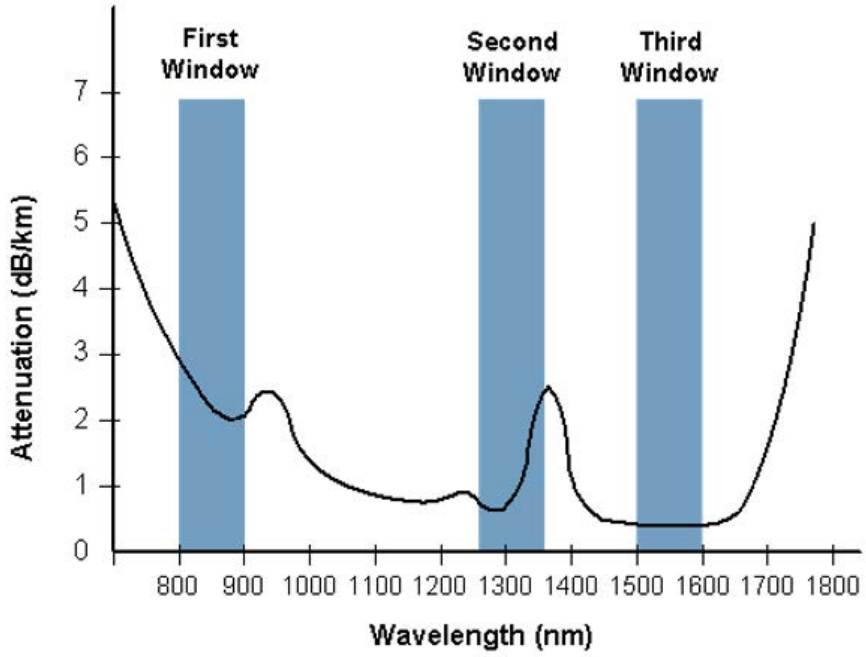


Figure 2.5: Attenuation curve as a function of wavelength and transmission windows for an optical fibre [9].

	Window Range
First Window	800 nm - 900 nm
Second Window	1260 nm - 1360 nm
Third Window	1500 nm - 1600 nm

Table 2.1: Transmission windows ranges of the optical fibre.

Band	Description	Wavelength Range
O band	original	1260 nm - 1360 nm
E band	extended	1360 nm - 1460 nm
S band	short wavelengths	1460 nm - 1530 nm
C band	conventional (”erbium window”)	1530 nm - 1565 nm
L band	long wavelengths	1565 nm - 1625 nm
U band	ultralong wavelengths	1625 nm - 1675 nm

Table 2.2: Further subdivision of the optical transmission windows.

The losses for the propagation of the light in fibres is 0.2 dB/km, and therefore amplifiers are not required up to tens of kilometres and the fibres are preferred over electrical cables when high bandwidth, long distance and immunity to electromagnetic interference are required.

Before elaborating on the problems that must be faced while working with optical fibre, in the next section a brief overview of optical perfomance evaluation is given.

2.3 Evaluation of System Performance

A variety of measurements can be carried out on optical communication systems in order to evaluate their performance. In this section, the most significant measurements are briefly described.

2.3.1 Bit Error Rate

The bit error rate (BER) is defined as the number of received bits which have been changed during the transmission due to the effects of noise or distortion. For instance, in the case of NRZ modulation in which there are only two levels of energy, corresponding to bit 0 and bit 1, the bit 0 may become a 1 or viceversa. A typical BER curve for DML is depicted in Figure 2.6.

The figure shows the BER versus received power of an optical transmission at 0 and 30 km. As you can see, when the transmission is back-to-back, the BER is lower than 10^{-3} for almost all the received power values taken into account. As soon as the transmission length is increased (30 km in the graph), the BER degrades (in this case of more than two orders of magnitude for the same received power).

Two types of BER can be distinguished: counting BER and stochastic BER. Counting BER is the simplest method and consists of comparing the received bits with the originally transmitted data. In this case, it is expressed as the ratio between the number of errors and the number of received bits:

$$\text{BER} = \frac{\text{number of errors}}{\text{number of received bits}} \tag{2.2}$$

The stochastic BER is the overall probability of erroneous identification of the received bits and it is estimated using the probability density function (PDF) of received symbols. The threshold level used to distinguish if the received bit is either a “0” or a “1” is optimized

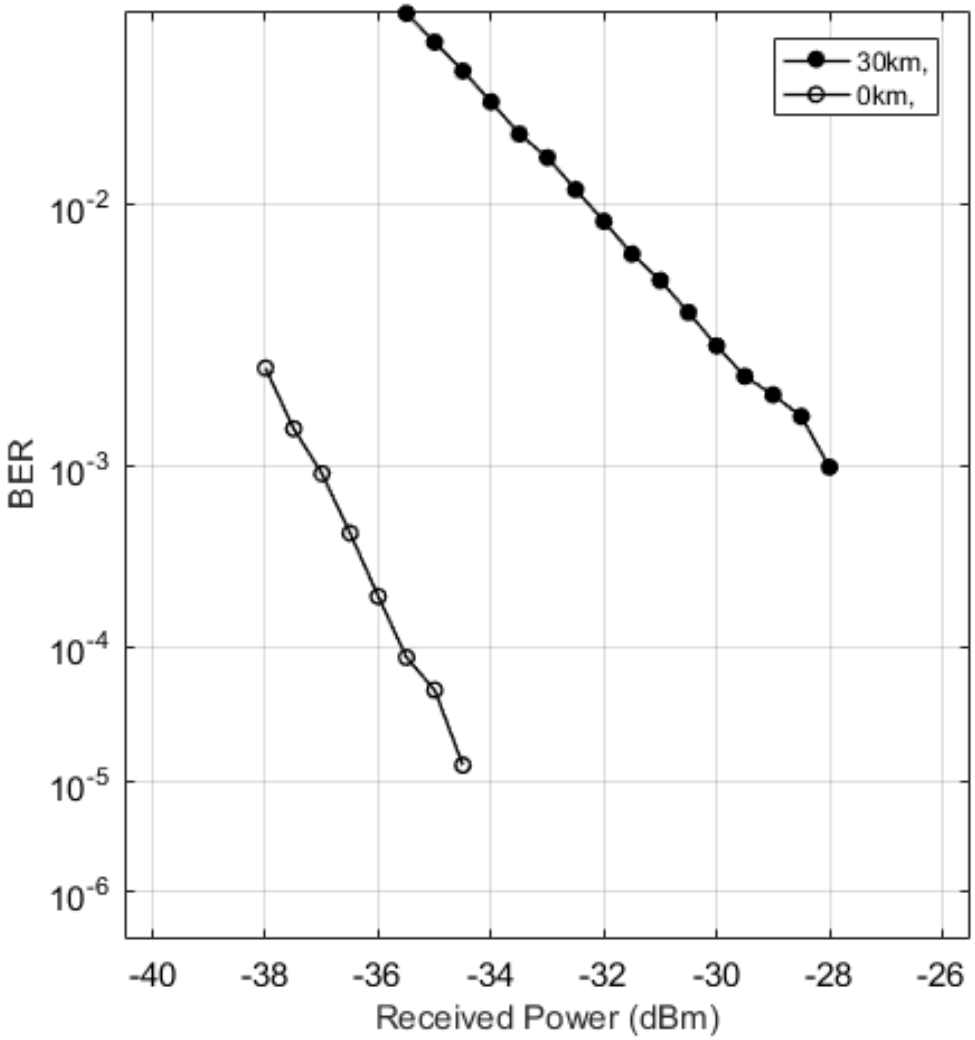


Figure 2.6: BER curve for two transmission lengths

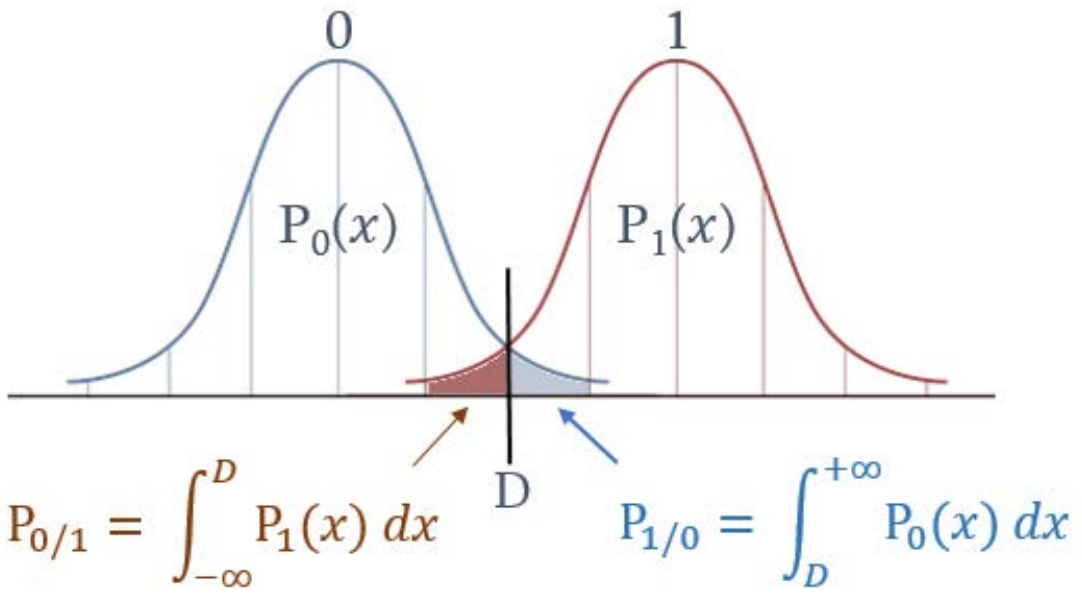


Figure 2.7: Gaussian distribution of the symbols and bit error probability.

to minimize the bit error rate.

After collecting a statistically representative number of samples for each symbol (let's say greater than 1000 samples per symbol), mean μ and variance σ^2 can be computed and, assuming the distribution is gaussian, the PDF (symbol probability) can be written as

$$P_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)} \tag{2.3}$$

where x is the symbol. Now, the probability of error for each bit can be computed as

$$P_{x/y} = \int_{-\infty}^{D} P_y(x)dx \tag{2.4}$$

or

$$P_{y/x} = \int_{D}^{+\infty} P_x(x)dx \tag{2.5}$$

depending on the position of the gaussian curve and where $P_{x/y}$ denotes the probability of receiving x given that the sent symbol is y. An illustration is given in Figure 2.7.

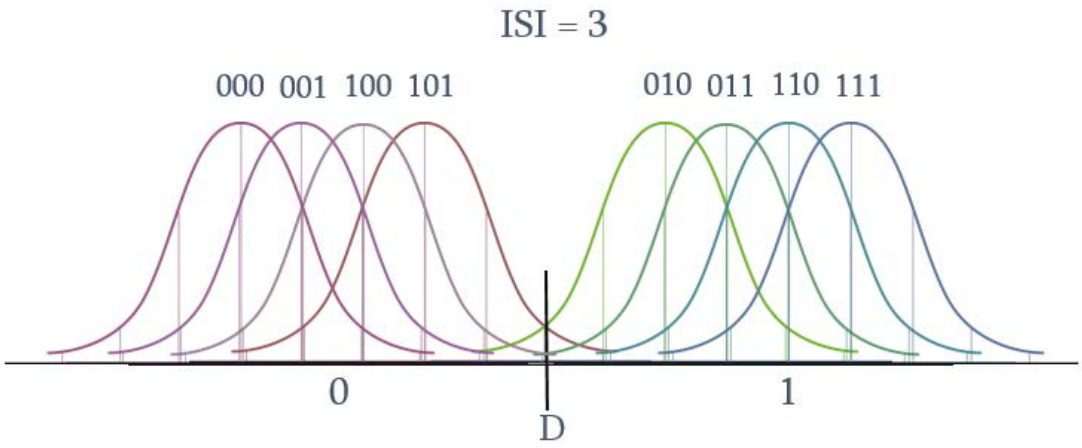


Figure 2.8: Gaussian distribution of the symbols with intersymbol interference.

When the symbol error probability is computed, the BER can be obtained as

$$\text{BER} = \sum_{x \in S} P_x \sum_{y \in S \setminus \{x\}} P_{y/x} \tag{2.6}$$

where S is the set of symbols. In the presence of *intersymbol interference* (ISI), the samples are not independent random variables because it introduces a correlation between adjacent bits. If ISI = 3 is taken into account, with one bit preceding and one bit succeeding that affects the current bit, the typical distribution is shown in Figure 2.8.

The BER can be computed using Equation 2.6.

2.3.2 Eye diagram

The eye diagram is a visual representation used to analyze a high-speed digital signal, by capturing some key parameters of the quality of the signal. The eye diagram is built from a waveform by overlapping the parts of the waveform corresponding to each bit into a single graph. The horizontal axis corresponds to the time, while the vertical axis corresponds to the amplitude of the signal. Figure 2.9 (a) shows an open eye diagram, captured by an oscillope, which represents a good transmission.

Some important measurements of an eye diagram are defined as follows (see Figure 2.10):

1. One level: represents the mean value of a logic '1'.
2. Zero level: represents the mean value of a logic '0'.

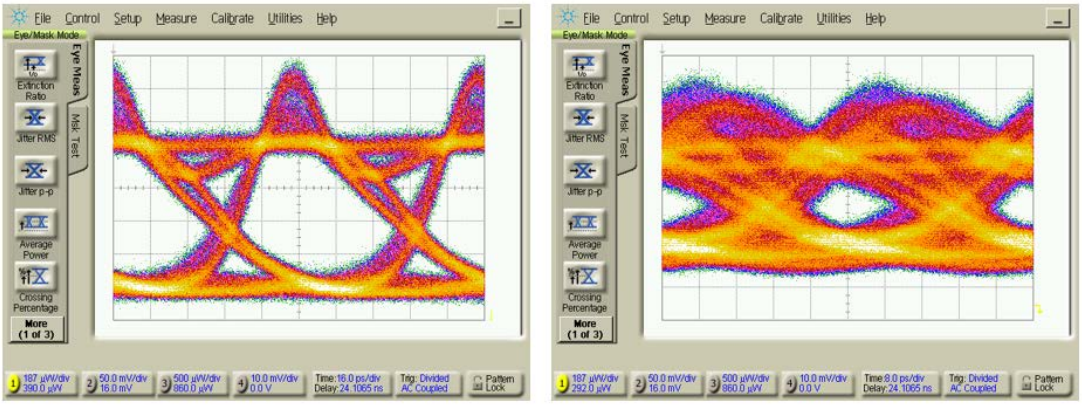


Figure 2.9: Eye diagrams: (a) open eye diagram; (b) very distorted eye diagram. The eye diagram are captured using an oscilloscope, by sending PRBS16 with NRZ at 12.5 and 25 Gb/s. The time division is 16 ps/div.

3. Eye amplitude: is the difference between the one and zero levels.
4. Eye height: is a measure of the vertical opening of an eye diagram. It determines the eye closure due to noise.
5. Eye width: is a measure of the horizontal opening of an eye diagram. It is measured as the difference between the statistical mean of the crossing points of the eye.

The opening of the eye diagram, combined with the BER, gives a suitable method of evaluating the performance of the transmission. Due to impairments and distortions caused by chromatic dispersion, noise, and intersymbol interference in the optical fibre, the eye diagram can appear closed. This is why the opening of the eye pattern can be used to quickly understand the quality of the transmission. An example of a closed eye diagram due to the distortion is shown in Figure 2.9 (b).

2.4 Laser Diodes and Directly Modulated Laser

2.4.1 LASER

The word *laser* is an acronym for Light Amplification by Stimulated Emission of Radiation and denotes the way electrons are stimulated in order to emit photons. When an electron absorbs a photon, it jumps to a higher energy level (called *absorption*), but the nature of the atom means that electrons want to fill the lowest energy states possible. When an electron drops from a higher state to a lower state it emits a photon, called *spontaneous emission*.

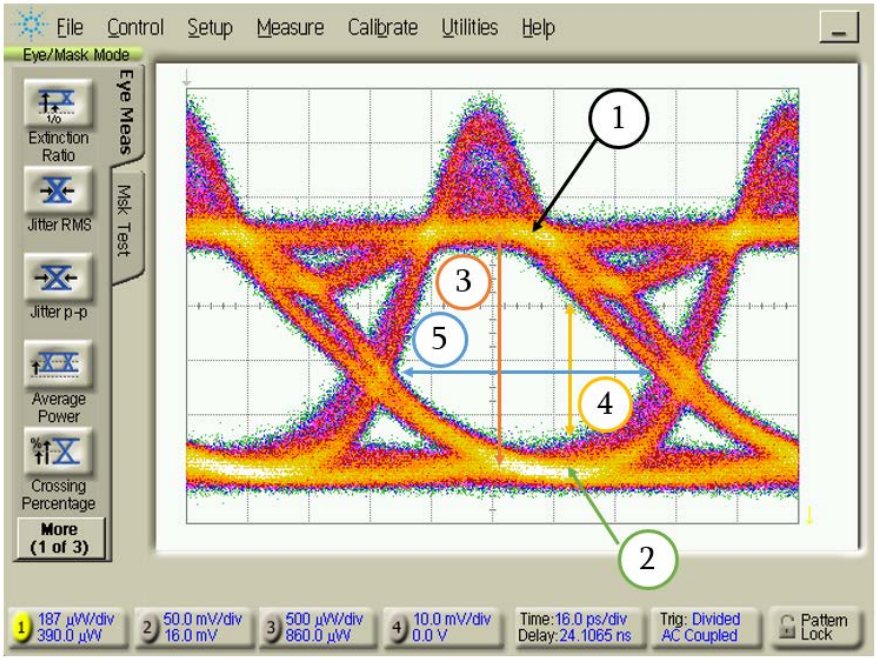


Figure 2.10: Measurement on eye diagram.

In order to get an electron to drop from a high state to a low one, it must interact with an electromagnetic wave or another photon. The aim is to produce photons which are in phase, of the same frequency and travelling in the same direction, characteristics of laser light. This can be done by the so-called *stimulated emission*, discovered by Albert Einstein in 1916, while he was investigating spontaneous emission.

When an electron in excited state interacts with a photon with specific energy, the electron falls back to a low energy state, emitting another photon with phase, frequency and direction equal to those of the photon with which it interacted. Moreover, many electrons in the excited state are needed to generate lots of photons, a process called *population inversion*. The *gain medium* of the laser, the material to which energy is supplied, has atoms which allow stimulated emission and population inversion to occur.

Finally, on both side of the gain medium, there are two mirrors, one of which is fully reflective, while the other allows some photons to pass through it in order to produce the laser output. Figure 2.11 shows the optical spectrum of the laser diode, captured using an Optical Spectrum Analyzer (OSA). The peak with the highest optical power corresponds to the wavelength of the photons that are allowed to pass by the mirror to generate the laser beam. The numerous peaks with lower powers correspond to the longitudinal modes in the

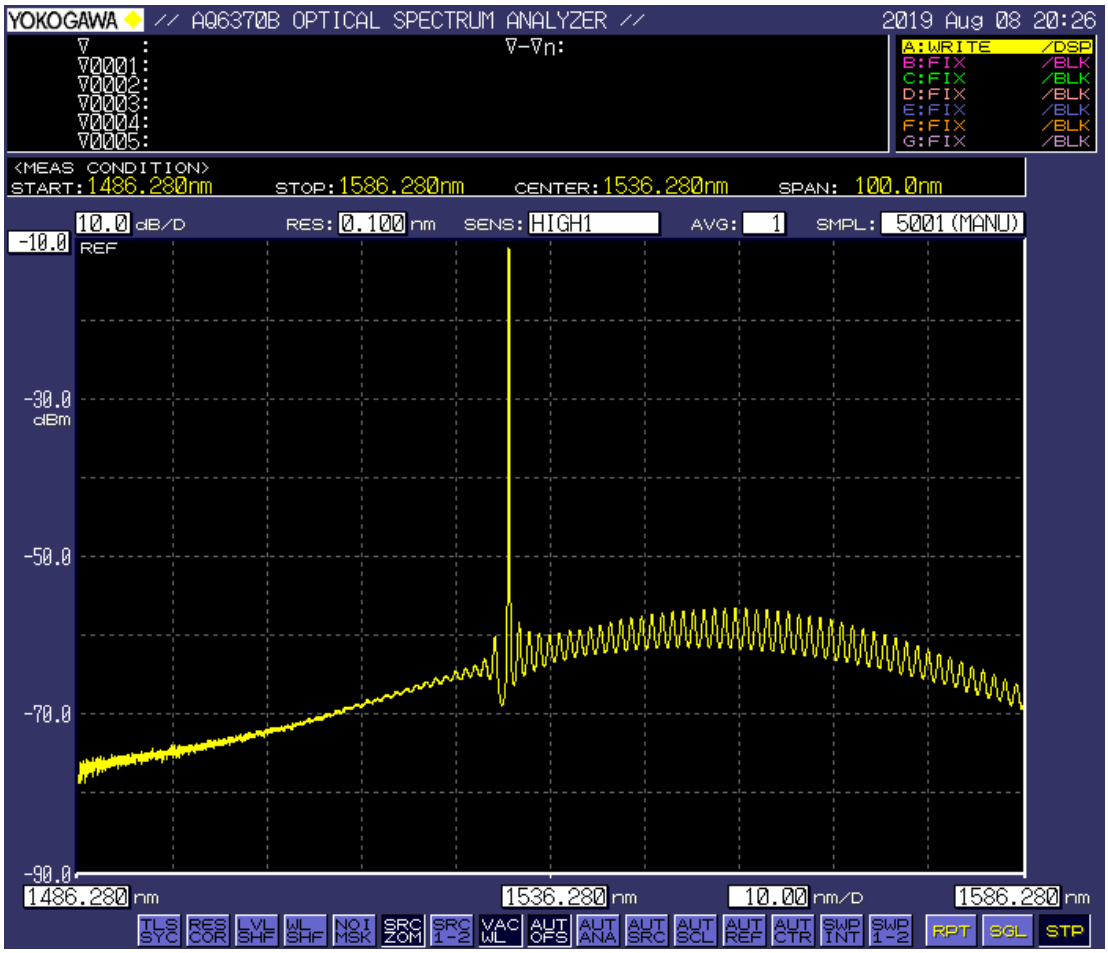


Figure 2.11: Distribution of the power od the laser diode captures with an Optical Spectrum Analyzer.

laser. Only specific frequencies (or wavelengths) are possible inside the optical cavity of a laser. The photons at those frequencies are blocked by the partially reflective mirror and this is why their optical power is lower.

2.4.2 Directly Modulated Laser

Despite the fact the term laser refers to the principle of operation, nowadays this word is used for devices which generate light using the laser principle. Laser technology has a central role in the area of optical communications because the laser beam can propagate over long transmission lengths without much divergence and can be emitted both continuously or in the form of short pulses.

One of the main components of an optical transmission system is the modulation opera-

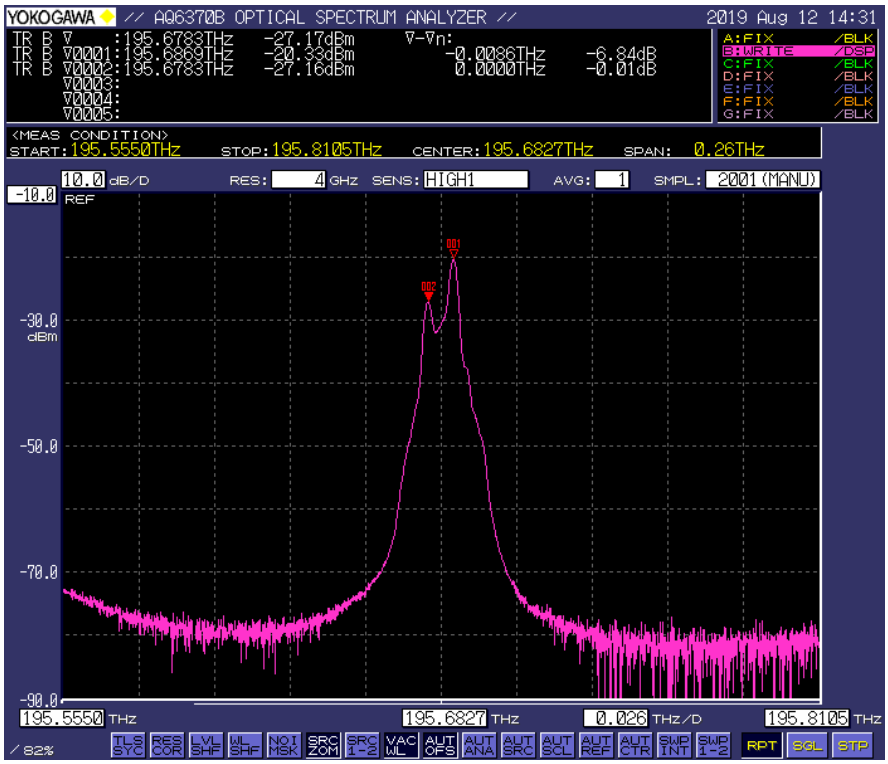


Figure 2.12: Optical spectrum of DML.

tion, which consists in imprints an electrical digital signal with high bitrate onto an optical carrier, such as a laser beam.

The simplest and the cheapest technique to modulate the lightwave is called *intensity modulation* in which the intensity of the light varies according to the data to be transmitted and can be easily detected by the photodiode at the receiver, depending on the incoming optical power. In particular, in the case of *direct modulation* with NRZ modulation, the light is emitted from a semiconductor laser with a higher power when a "1" is transmitted, while when a "0" is transmitted the power is lower. Figure 2.12 shows the optical spectrum of a DML captures through the OSA. The two peaks correspond the optical power corresponding to levels "0" and "1" which are at different frequencies (or wavelength).

A typical characteristic curve of a DML diode is shown in Figure 2.13. The characteristic has been taken experimentally and it shows the laser diode's output optical power versus injected electrical current. The output optical power varies as a function of the current passing through the diode. Below the threshold current, the output power is very low, and due to spontaneous emission, it is not useful for optical communication applications. But as the

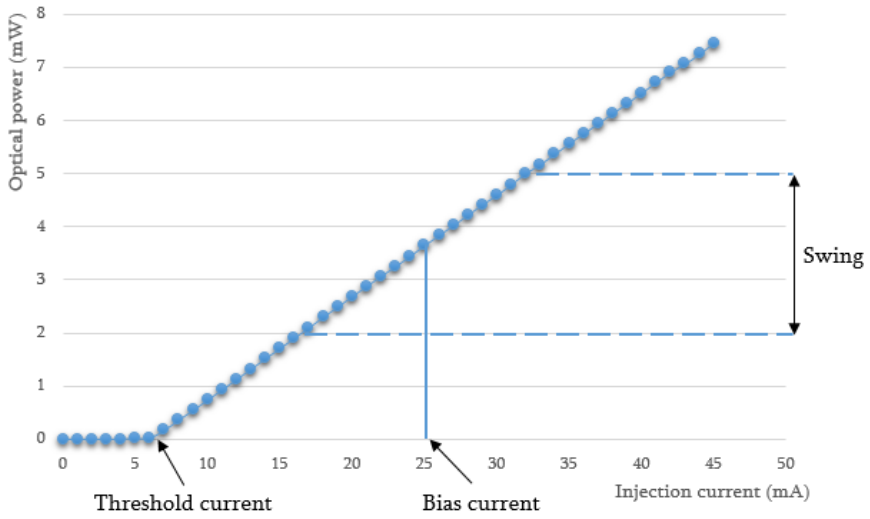


Figure 2.13: Directly Modulated Laser operational characteristic.

current increases over the threshold, the output optical power increases significantly with a sharp slope. In this region, the laser is in lasing mode with stimulated emission of photons. This is a useful area in which the laser diode can be exploited. In particular, to choose the region of operation of the DML, the bias current, that represents the DC current applied to laser and the swing, that represents the amplitude of interval between the lowest and highest level of power around the bias current must be set. This characteristic curve strongly depends on temperature. The higher the temperature, the higher the threshold current.

Figure 2.14 shows the principle of operation of a DML. The drive current varies according to the signal to be transmitted and the optical power is proportional to the injection current.

Extinction Ratio

An important parameter of DML is the *extinction ratio* (ER), defined as

$$ER = \frac{P_1}{P_0} \tag{2.7}$$

where P_1 and P_0 are the power levels corresponding to bits "1" and "0", respectively. In order to achieve a large separation between the power of "1" and "0", a high extinction ratio is required. On the other hand, with a poor extinction ratio, higher optical power at the receiver is required to achieve a given bit error rate (BER). The reasons will be clear in the next sections.

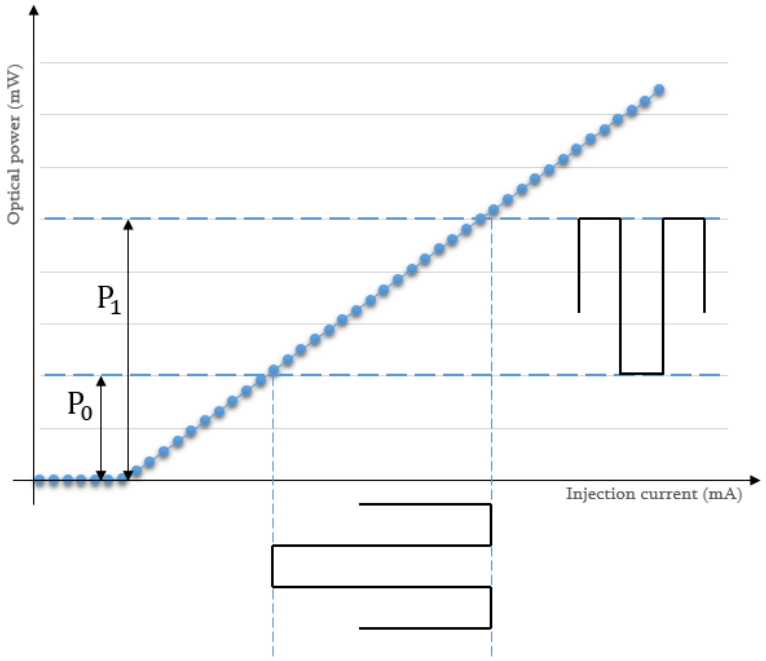


Figure 2.14: Direct current modulation of a semiconductor laser.

Chromatic Dispersion

A 10-Gbit/s laser is able to transmit 10 billion bits every second, meaning that each of these bits can last a maximum of 100 ps. After travelling through several kilometres of optical fibre, it could happen that the pulses are spread out in time. This means that the 100 ps duration pulses could now be 150 ps or 200 ps at the receiver,making it difficult to recognize the transmitted bits at the receiver. The problem becomes harder for increased bit rates of the transmission. This phenomenon is known as dispersion.

One form of dispersion occurs because different wavelengths of light travel through the optical fibre at different speeds. The refractive index of the core of the fibre varies with wavelength, causing the wavelengths to travel at different speeds and this effect is called *material dispersion*. The geometry and refractive index profile of the waveguide causes the wavelengths to travel at different speeds and this is known as *waveguide dispersion*. The combination of the material dispersion and the waveguide dispersion is called *chromatic dispersion* (CD). The effect of the CD on a transmission of pulses through the optical fibre with different bitrate is depicted in Figure 2.15. The delay that occurs between the subsequent symbols leads to ISI.

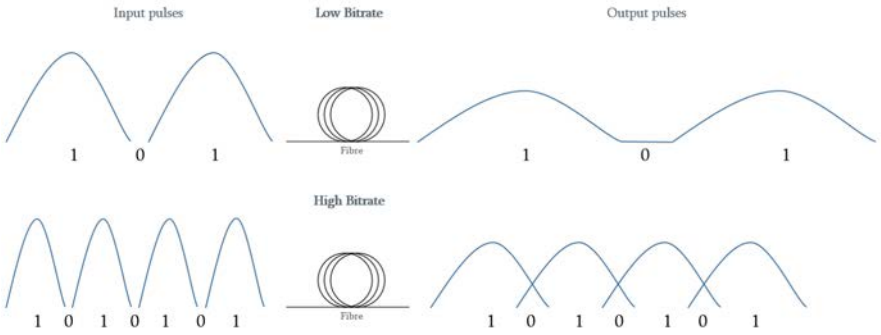


Figure 2.15: Effect of chromatic dispersion with low and high bitrate.

Chromatic dispersion is measured in ps/nm/km, meaning that a pulse with a wavelength of 1 nm, will disperse by 1 ps for each kilometre of fibre. Chromatic dispersion becomes more serious at higher bit rates and degrades the system performance in such a way that it limits the transmission distance in the optical fibre. For example, with a 10-Gbit/s laser after 500 km transmission length, even with a small chromatic dispersion, the bit values at the receiver are completely indistinguishable. The effect of chromatic dispersion depends on the transmission length and the bitrate; by increasing one of these two quantities (that means increasing the BL product) the effect of the chromatic dispersion becomes an issue.

Chirp

The *chirp* of an optical pulse is a signal in which the instantaneous frequency varies with time. An up-chirp means that the instantaneous frequency rises with time, and a down-chirp means that the instantaneous frequency decreases with time. The chirp can limit the performance of an optical fibre system, especially when the laser is directly modulated because the desired intensity modulation of the lightwave is accompanied by a modulation of its phase. When the laser is directly modulated, a change in the bias current will lead to a change in the carrier density and, as a consequence, this will lead to a change in the refractive index of the gain material. Since the wavelength of the laser is influenced by the gain material, the instantaneous frequency of the emitted signal will be time-varying. This means that directly modulated lasers are inherently chirped.

The electric field of a lightwave can be expressed as

$$E(t) = \Re[A(t)e^{j\omega_0 t}] \tag{2.8}$$

where ω_0 is the carrier angular frequency, $\Re$ denotes the real part and $A(t)$ is the complex envelope of the signal which can be written as

$$A(t) = |A(t)|e^{j\phi(t)} = \sqrt{P(t)}e^{j\phi(t)} \tag{2.9}$$

where $P(t)$ is the power of the signal. The instantaneous frequency of the optical signal can be expressed as

$$\omega(t) = \omega_0 + \frac{\partial\phi}{\partial t} \tag{2.10}$$

In the case of directly modulated laser, the information pulses will spread out, leading to ISI which will introduce errors at the receiver, increasing the bit error rate.

The variations of the instantaneous frequency at the output of the DML can be expressed as

$$\Delta\upsilon(t) = \frac{\alpha}{4\pi}\left(\frac{d}{dt}[\log P(t)] + \kappa P(t)\right) \tag{2.11}$$

where α is the *linewidth enhancement factor*, a parameter quantifying the amplitude-phase coupling in a laser, and κ is the *adiabatic chirp coefficient*.

Therefore the chirp consists of two terms. The first term, called *transient chirp*, appears only when the input power of the laser varies with time. The second term, called *adiabatic chirp*, is responsible for the different frequencies observable when a "0" or a "1" is transmitted. Figure 2.16 shows the simulation of the waveform and the corresponding chirp at the output of the directly modulated laser, while Figure 2.17 shows the ouput of the DML (without fibre) after the transmission of a pseudorandom binary sequence (PRBS) with NRZ modulation at 12.5 Gb/s, captured by an oscilloscope.

By increasing the bias current the effect of the transient chirp can be reduced. Figure 2.18 shows the effect of the chirp with a bias current of 21 mA (Figure 2.18 (a)) and 25 mA (Figure 2.18 (b)) on a pulse fo 300 ps. On the other hand, this leads the ER to reduce because with high bias current the swing cannot be too high for not operating in the saturation region of the DML.

Chromatic dispersion combined with DML chirp can seriously degrade a signal. Therefore, larger transmission distances require a low chirp. The aim of this dissertation is to generate a pre-distorted signal to be transmitted using a directly modulated laser which precompensates the effects of the chromatic dispersion, leading to a signal which is not as degraded.

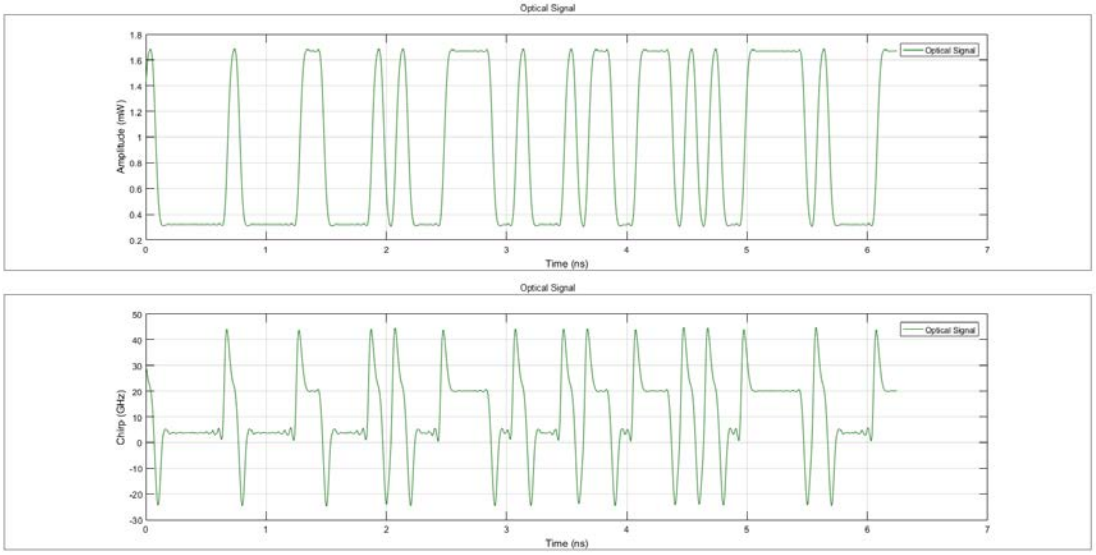


Figure 2.16: Simulated waveform and chirp at the output of a directly modulated laser.

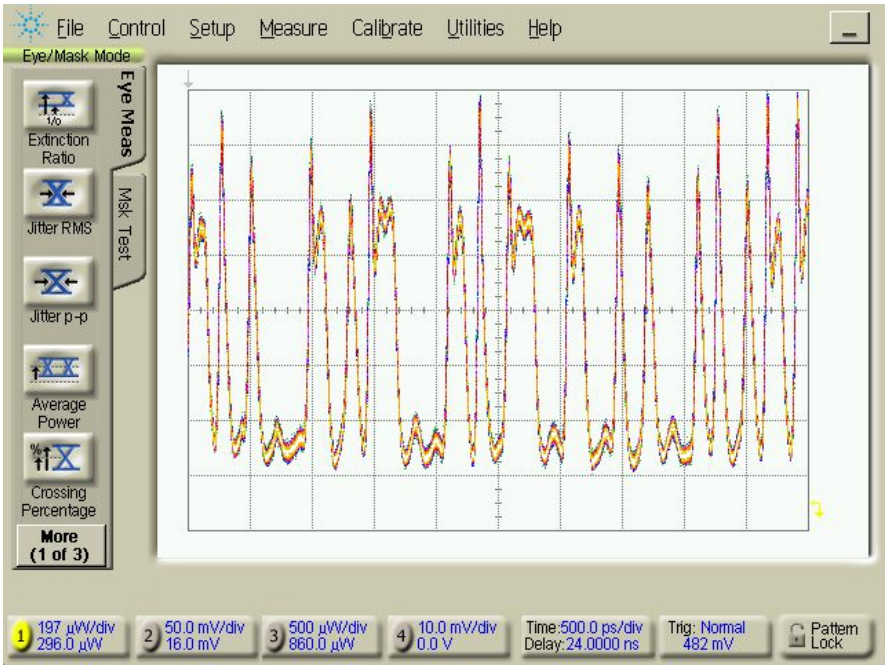
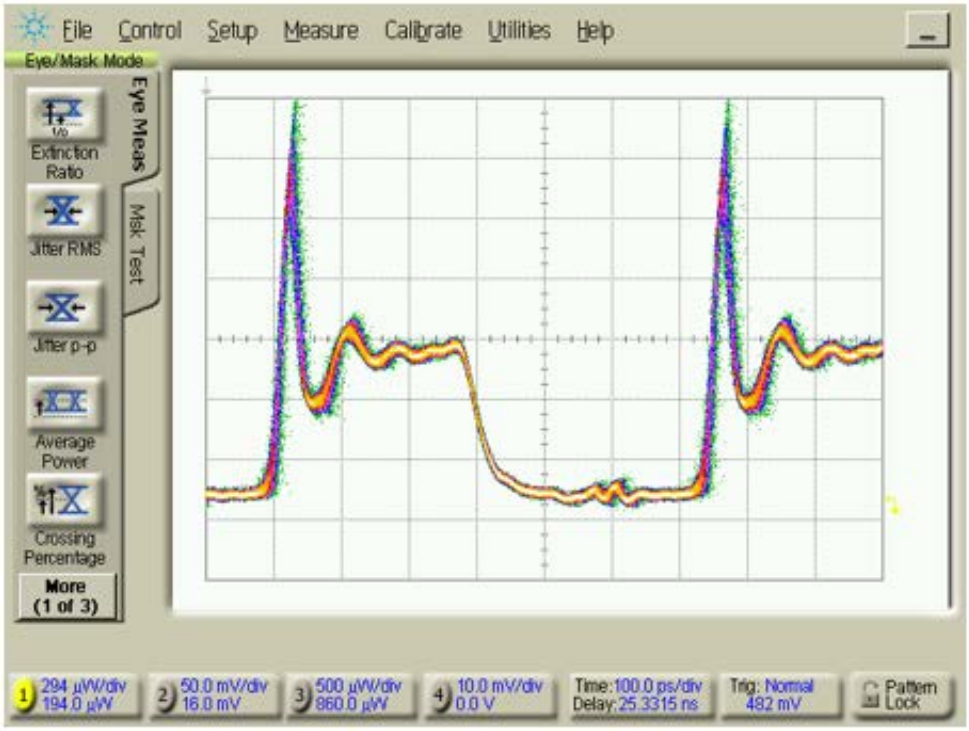


Figure 2.17: Effect of the chirp on the transmission of a sequence with NRZ modulation of PRBS15 at 12.5 Gb/s.

(a)



(b)

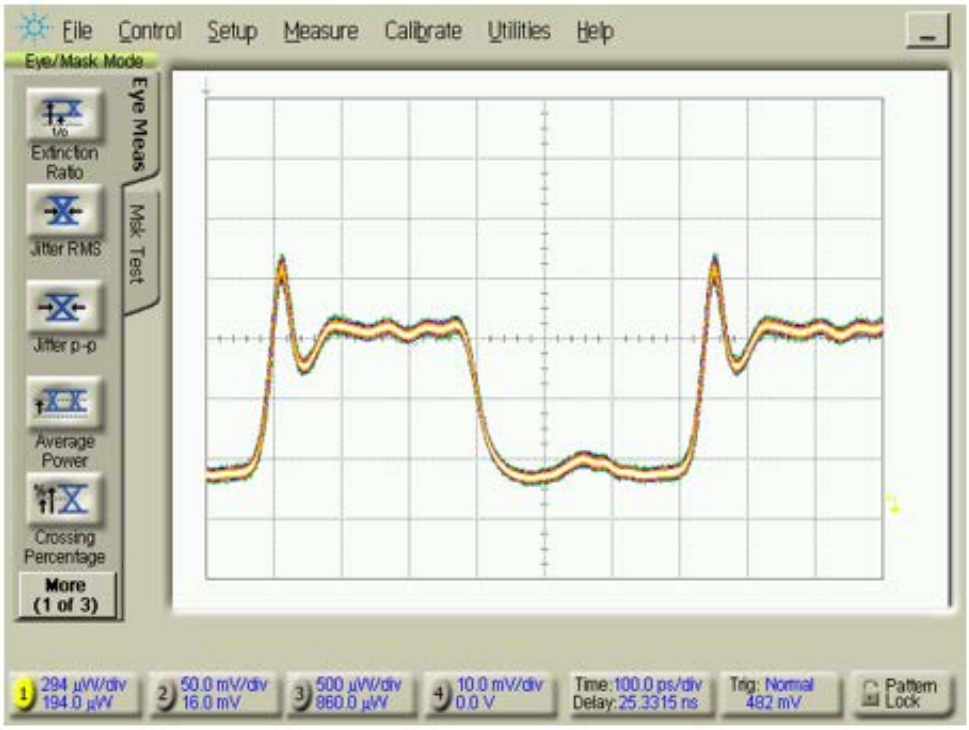


Figure 2.18: Effect of the transient chirp (overshoot) at the output of a directly modulated laser with bias current at (a) 21 mA and (b) 25 mA.

The usual techniques exploit special pieces of hardware, like the Mach-Zehnder Modulator (MZM) which allows the signal to be decoupled in magnitude and phase and that can be considered as an ideal transmitter, or chirp managed laser (CML) which applies a filter at the transmitter in order to produce high extinction ratio pulses. The proposed solution, instead, is software-based, exploiting of advanced digital signal processing neural networks and undirect learning algorithms as forms.

3

Artificial Neural Networks

An artificial neural network is a model of computation inspired by the neuronal structure of human brain. As a simpler model of the brain, it consists of a large number of basic computing particles called neurons that are interconnected in a complex communication network and where the output of any given neuron may be the input to thousands of other neurons.

Learning with neural networks was proposed in the mid-20th century in attempts to find a mathematical representation of information processing in biological systems [11] [12] [13] [14]. It yields an effective learning paradigm and has recently been shown to achieve cutting-edge performance on several learning tasks [15] [16] [17] [18].

3.1 Feedforward Neural Networks

A neural network can be described as a directed graph whose nodes correspond to artificial neurons and edges correspond to links between them. The term feedforward refers to the fact that the graph is directed and that does not contain any cycles. Each neuron receives as input a weighted sum of the values of the neurons connected to it. A fuction called *activation function* is applied to the weighted sum in order to introduce non-linearity. Indeed, each neuron can be modeled as a scalar function $\sigma : \mathbb{R} \to \mathbb{R}$.

The most common activation functions used in neural networks are *sigmoid function*, *tanh*, *ReLU function* (Rectified Linear Unit function) and *Leaky ReLU* whose equations

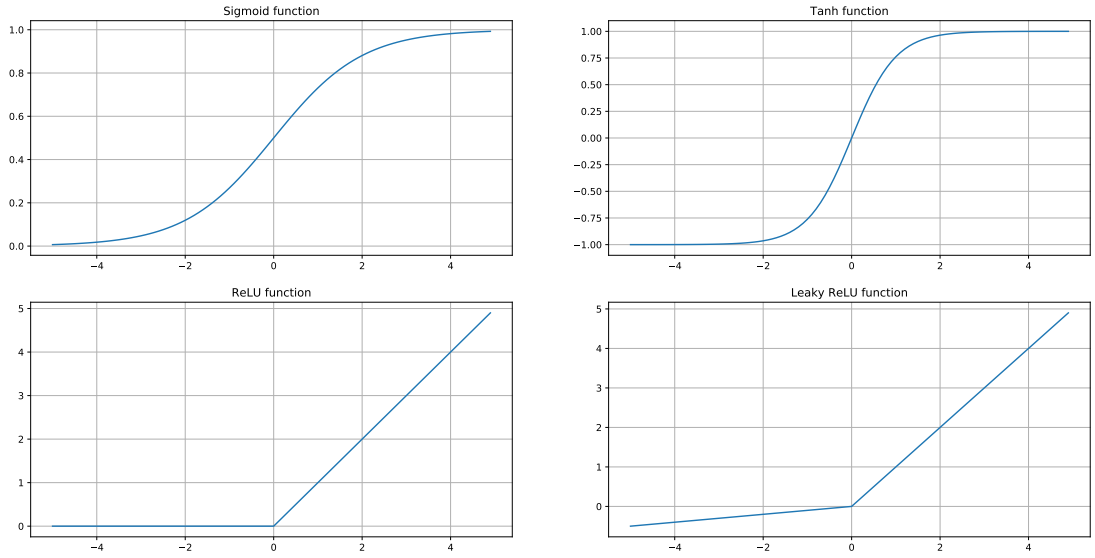


Figure 3.1: Plots of the activation functions.

are:

$$f(x) = \frac{1}{1+e^{-x}} \qquad \text{sigmoid} \tag{3.1}$$

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad \tanh \tag{3.2}$$

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \qquad \text{ReLU} \tag{3.3}$$

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \qquad \text{Leaky ReLU} \tag{3.4}$$

The plots of the functions are shown in Figure 3.1.

Neurons are connected in hierarchical networks, with the output of some neurons being the inputs to others. Each node takes multiple weighted inputs that are summed together and the activation function is applied to the summation, generating its output. We can represent this network as a union of disjoint subsets of nodes in which each subset constitutes a layer of the network, such that every edge of a layer connects one node to the neuron in the following layer (see Figure 3.2). These edges are called *weights*.

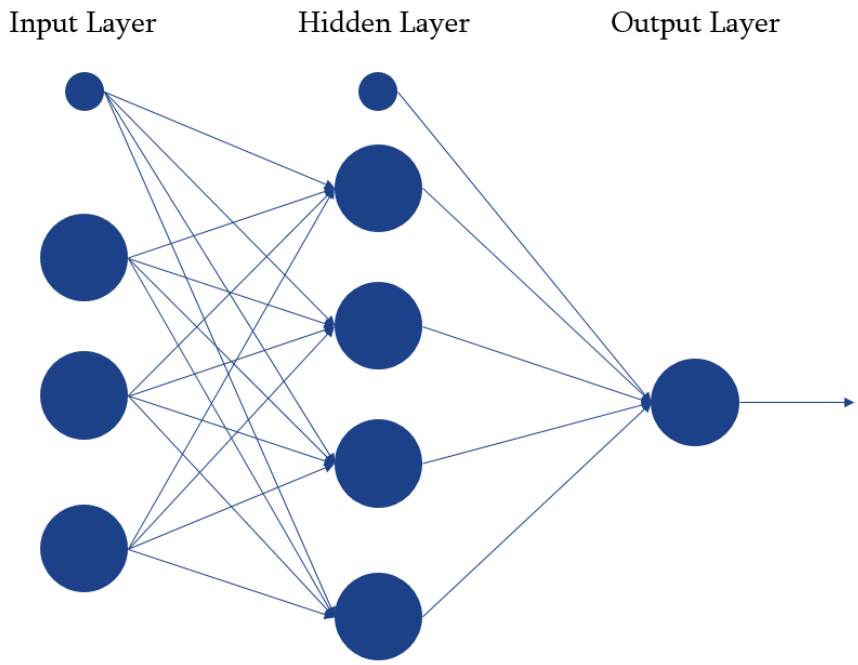


Figure 3.2: Architecture of a 2-layers neural network.

The first layer is called the *input layer* and it contains n+1 neurons, where n is the dimensionality of the input space, while the last neuron is a neuron whose value is always 1 and it is called *bias* and it is multiplied by its weight. The last layer is called the *output layer* whose number of neurons depends on the dimensionality of the output of the network. The layers in the middle are called *hidden layers*. If the neural network is composed by $K-1$ hidden layers, it is called K layers neural network and K is called the depth of the network. Figure 3.2 shows a 2-layers neural network with one output neuron.

Indeed, the output of each neuron is a nonlinear function of a linear combination of the inputs. This linear combination z_j of the j-th neuron can be expressed as

$$z_j^{(\ell)} = \sum_{i=1}^{n^{(\ell)}} w_{ji}^{(\ell)} x_i + w_{j0}^{(\ell)} \tag{3.5}$$

where the superscript (ℓ) indicates the parameters of the ℓ-th layer of the network, $n^{(\ell)}$ is the number of input neurons to j-th neuron in the ℓ-th layer, $w_{ji}^{(\ell)}$ are the *weights* and $w_{j0}^{(\ell)}$ is the bias. A nonlinear activation function $f^{\ell}(\cdot)$ in the ℓ-th layer is applied to this linear

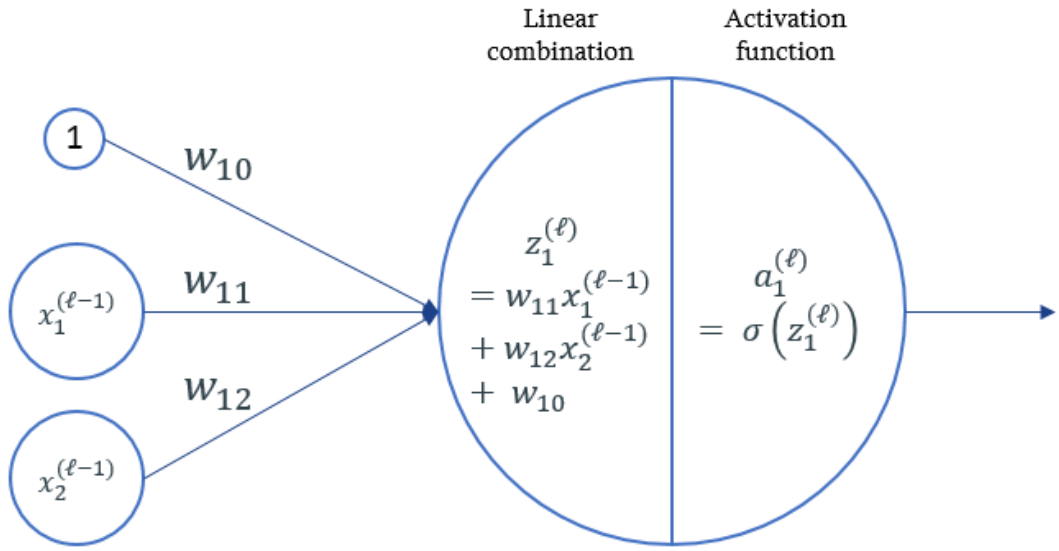


Figure 3.3: Forward propagation diagram.

combination which is the output of the j-th neuron

$$a_j^{(\ell)} = f^{(\ell)}(z_j^{(\ell)}) \tag{3.6}$$

Figure 3.3 shows a diagram of the forward propagation step.

The choice of the activation function to apply in the output layer depends on the nature of the data and on the target variables. For instance, if the neural network is used for a binary classification task the most suitable activation function is the sigmoid function, while for a regression task a linear function in the output layer could be the one which works best.

The process of evaluating Equations 3.5 and 3.6 through all the layers of the network can be considered as a forward propagation of information through the network.

The output of the k-th neuron in the following layer will be

$$a_k^{(\ell+1)} = f^{(\ell+1)}\left(\sum_{j=0}^{n^{(\ell+1)}} w_{kj}^{(\ell+1)} f^{(\ell)}\left(\sum_{i=0}^{n^{(\ell)}} w_{ji}^{(\ell)} x_i\right)\right) \tag{3.7}$$

where the bias term is absorbed into the set of weights by defining the variable $x_0 = 1$.

This processing is also known as *multilayer perceptron* or MLP. This name could seem confusing because it refers to *perceptron neurons*, proposed by the scientist Frank Rosenblatt in the 1950s and 1960s and inspired by earlier work by McCulloch and Pitts, in which it takes

several binary inputs and it produces a single binary output, depending on a threshold value

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases} \tag{3.8}$$

While in MLP networks inputs and outputs can be any values in $\mathbb{R}$.

The approximation properties of feedforward networks have been widely studied and found to be very general. This is the reason why neural networks are said to be *universal approximators*. There is an extremely low *inductive-bias* in neural networks because they do not require much feature engineering. For example, a neural network can be used to fit any continuous function. This will be shown in the next sections.

3.2 Neural Network Training

By propagating the input through all layers of the network, a value or a set of values (depending on the size of the output layer) called prediction is obtained. Training a neural network involves adjusting the weight parameters of all neurons so as to optimise its output with respect to some training data. In supervised learning, that means that the target values are known, the training is performed by reducing the error between the prediction, after the propagation of the input, and the desired output.

The concept of supervised learning is to provide many input-output pairs of known data and modify the weights of the network in order to minimize the error. These input-output pairs can be minimised as $\{(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^m, \mathbf{y}^m)\}$ where m is the number of training samples in the dataset.

To quantify how well the prediction and the target value(s) are close we define a *cost function*, sometimes referred to as *loss function*.

3.2.1 Cost function

Mean Squared Error

The mean square error or MSE or quadratic loss is measured as the average of squared difference between predictions and target values. It is non-negative and due to squaring, predictions which are not close to the target values are much more penalised. It is defined as

follows

$$\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} \parallel y_i - a_i \parallel^2 \tag{3.9}$$

where m is the number of training samples, a_i is the prediction of the i-th training sample and y_i is the target value corresponding to the i-th training sample. Often the factor 2 is put on the denominator in order to simplify the computation of the derivative, used in the algorithm used for adjusting the weights and biases of the network. MSE is usually used for regression tasks.

Cross-entropy

The cross-entropy cost function is usually used when the activation function of the output neuron is the sigmoid function. In particular, when there are only two possible targets, the *binary* cross-entropy is defined as

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log a_i + (1 - y_i) \log(1 - a_i)] \tag{3.10}$$

where m is the number of training samples, a_i is the prediction of the i-th training sample and y_i is the target value corresponding to the i-th training sample.

As the MSE, the cross-entropy is non-negative since the arguments of both the logarithms are in the range 0 to 1 and there is a minus sign before the sum.

Moreover, if the prediction is close to the target output for all training sample, then the cross-entropy will be close to zero. The cross-entropy is used for binary classification tasks.

However, the cross-entropy could be generalized for multi-class classification tasks, by summing over all the outputs in the output layer, as follows

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n^{(L)}} \left[y_i^j \log a_i^j + (1 - y_i^j) \log(1 - a_i^j) \right] \tag{3.11}$$

where m is the number of training samples, $n^{(L)}$ is the number of neurons in the output layer, a_i^j is the prediction of the i-th taining sample for the j-th neuron in the output layer and y_i^j os the target value corresponding to the i-th training sample for the j-th neuron in the output layer.

This is known as *categorical cross-entropy loss* or *softmax loss*, in which the output layer is usually a *softmax layer*. The softmax function is an activation function that turns numbers (called *logits*) into probabilities that sum to one and represents the probability distribution of a list of potential outcomes. It is defined as follow

$$f(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{C} e^{y_j}} \tag{3.12}$$

where C is the number of classes and that it corresponds to the number of neurons in the output layer.

3.2.2 Stochastic Gradient Descent and Backpropagation

Once the cost function to optimize is defined, weights and biases must be adjusted in order to have a prediction as close as possible to the target value for each input. This is a very general problem found in mathematical optimization and usually rely on an algorithm called *gradient descent*.

The gradient descent is a standard approach for minimizing a differentiable convex function $f(\mathbf{x})$. The gradient, denoted as $\nabla f(\mathbf{x})$, is the vector of partial derivatives of f. Gradient descent is an iterative algorithm in which at each iteration the current point is updated in the direction of the negative of the gradient. The update step is

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)}) \tag{3.13}$$

where $\eta > 0$ is a parameter called *step size* or *learning rate* which influences the convergence speed of the algorithm.

A learning rate which is too low causes a slow convergence, while if it is too high may overshoot the correct path towards the minimum of the function. Figure 3.4 shows an example of iteration of the algorithm. In each step the current point is moved in the opposite direction to the gradient and it approaches the minimum of the function step by step.

More precisely in the training of a NN, in the Equation 3.13 the function f is replaced by the used loss function C while the variable $\mathbf{x}$ is replaced by the weights and biases of the network, that is

$$\mathbf{W} := \mathbf{W} - \eta \cdot \frac{\partial C}{\partial \mathbf{W}} \tag{3.14}$$

where $\mathbf{W}$ is the matrix of weights which includes also the biases.

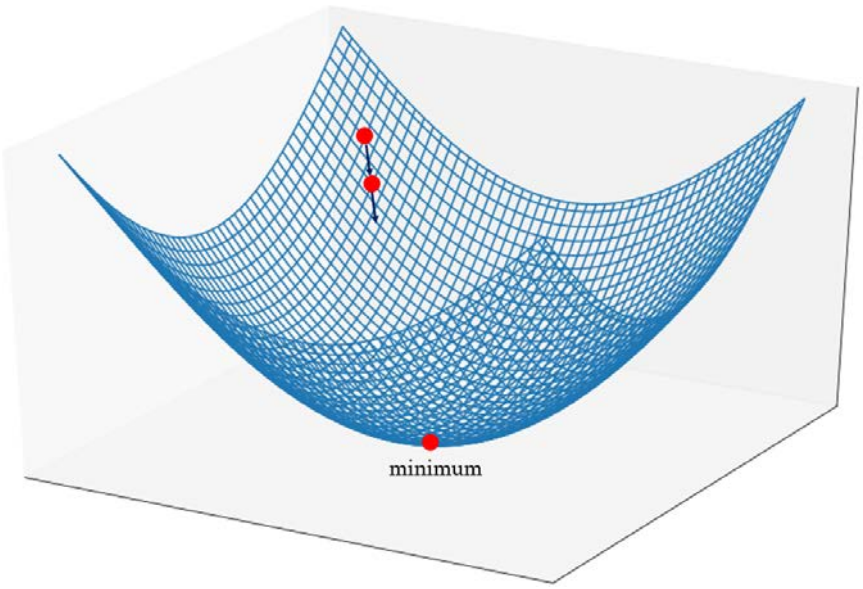


Figure 3.4: Example of gradient descent iteration.

The major problem of this algorithm is that it is very slow or not optimal because normally the gradient is computed over the entire training set. A neural netowrk may have thousands of parameters and this means that for each training sample in the dataset the algorithm requires thousands of operatins to evaluate. For this reason a modify version of the algorithm has been introduced, called *stochastic gradient descent* (SGD).

3.2.3 Gradient Descent variants

The aforementioned algorithm sometimes does not guarantee good convergence. For example, choosing a proper learning rate can be difficult. If it is too small the convergence can be too slow, while if it is too large it can cause the loss function to diverge or to fluctuate around the minimum.

Therefore, some variants which try to optimize the GD have been proposed [19] [20] [21].

Stochastic Gradient Descent

SGD does not require the update of the direction towards the minimum of the cost function to be based exactly on the gradient, but the direction become a random vector whose expected value at each iteration is equal to the gradient direction. With SGD, the gradient is computed with respect to a single training sample in the dataset and by performing a weight

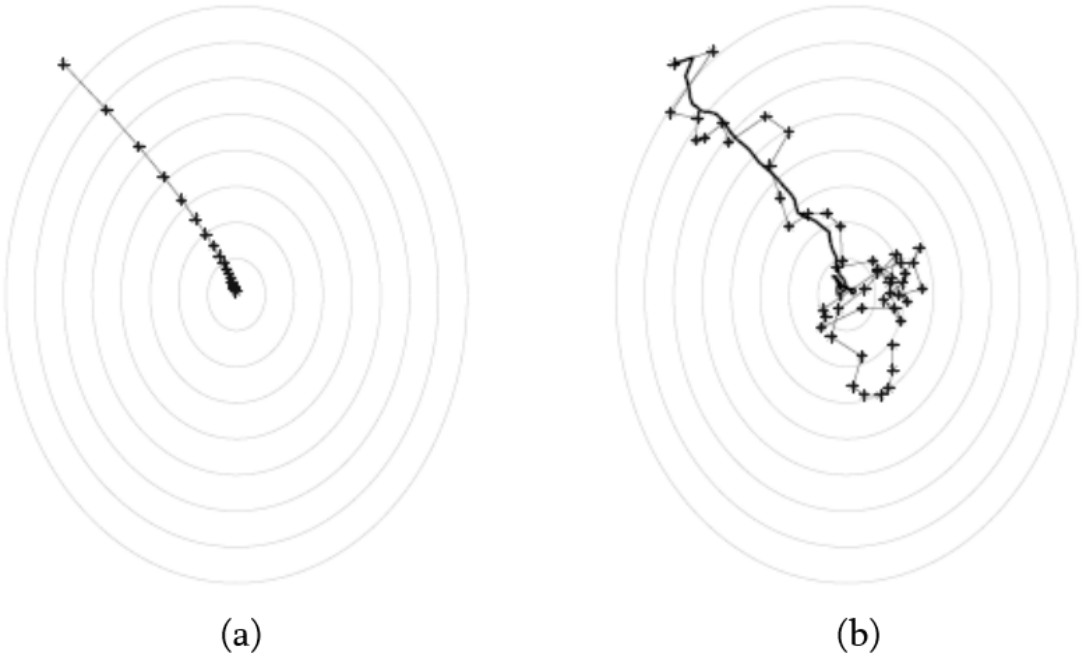


Figure 3.5: Comparison between (**a**) gradient descent and (**b**) stochastic gradient descent [15]

update for each. A comparison between the Gradient Descent and SGD in shown in Figure 3.5.

The pseudocode of SGD algorithm is listed in Algorithm 3.1.

Algorithm 3.1 Stochastic Gradient Descent for Neural Networks

Parameters :
number of iterations T, learning rate η
initialize the weights randomly and the biases to zero
for $i := 1$ to T
    sample a training sample in the training set
    compute the gradient ∇C through the backpropagation algorithm
    update $\mathbf{W} := \mathbf{W} - \eta \cdot \nabla C$

Each iteration of the for loop is called *epoch*. The number of epoch decides how many times the error is backpropagated in order to adhjust the weights of the network.

In each epoch, another routine called *backpropagation* algorithm is run. The backpropagation is the technique applied for evaluating efficiently the gradient of the cost function, by sending information alternately forwards and backwards through the network.

In order to evaluate the derivatives, the errors must be computed for each hidden and output neuron of the network. For example, for the ouput layer with sigmoid activation function f and binary cross-entropy cost function $\mathcal{L}$, the error δ_j can be computed as follows

$$\delta_j = y_j - a_j^{(L)} \tag{3.15}$$

For the hidden units, instead, the chain rule for partial derivatives can be exploited as follows

$$\delta_j = \frac{\partial \mathcal{L}}{\partial z_j} = \sum_k \frac{\partial \mathcal{L}}{\partial z_k} \frac{\partial z_k}{\partial z_j} = f'(z_j) \sum_k w_{kj} \delta_k \tag{3.16}$$

where $f'(\cdot)$ is the derivative of the activation function with respect to the j-th neuron. The pseudocode 3.2 lists the backpropagation algorithm.

Algorithm 3.2 Backpropagation

forward :
for $\ell := 1$ to L
    for $i := 1$ to $n^{(\ell)}$
        compute the output of each neuron according to Equations 3.5 and 3.6

baackward :
for $\ell := T - 1$ downto 1
    for $i := 1$ to $n^{(\ell)}$
        compute the partial derivative for each neuron according to Equations 3.15 and 3.16

Despite the fact SGD is faster if it is computed on a portion of training set and useful when the surface of the cost function is particularly irregular and when the training set is too big to be loaded into memory, the usual approach is called *mini-batch gradient descent*, in which the dataset used for the training is divided into a number of equally-sized mini-batches of K (usually a power of 2) samples each. This variant generalizes both gradient descent and SGD by setting K equal to m or 1, respectively.

Momentum

One variant of the algorithm is called *Gradient Descent with momentum*, proposed by Rumelhart, Hinton and Williams [19] and inspired by moving avarage technique. The weight up-

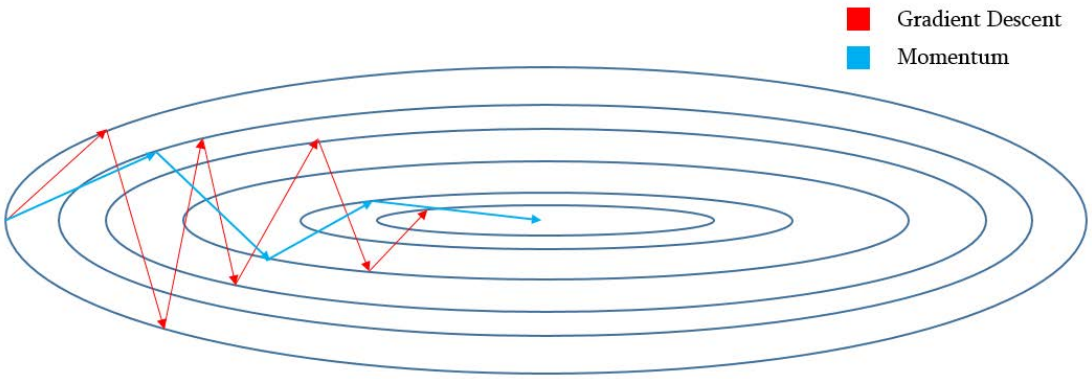


Figure 3.6: Comparison between gradient descent and momentum.

date is no longer a function of just the gradient at the current iteration, but it is a linear combination of the gradient and the previous update.

The equations for the update at the t-th iteration are

$$\mathbf{v}_t := \beta \mathbf{v}_{t-1} + (1-\beta)\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{t-1}} \tag{3.17}$$

$$\mathbf{W}_t := \mathbf{W}_{t-1} - \eta \mathbf{v}_t \tag{3.18}$$

where the subscript t and $t-1$ refers to the iteration number, β is a hyperparameter (usually 0.9 or similar value is used). At the first iteration, $\mathbf{v}_t$ is initialized to 0.

In this case, the weights are updated taking into account the gradient in the previous iterations. The epoch can consist of many iterations, depending on if a batch technique is used or not. Equations 3.14 and 3.18 coincide if β is set equal to 0.

This technique stems its name from the momentum in physics: it is similar to a ball pushed downhill and even if it is in a region in which the gradient changes considerably, it carries on going in the same direction, by reducing oscillation and gaining a faster convergence.

The comparison between GD and momentum is shown in Figure 3.6.

RMSProp

RMSProp which stands for Root Mean Square Propagation is adaptive learning rate method proposed by Geoff Hinton [20] as a variant of mini-batch gradient descent.

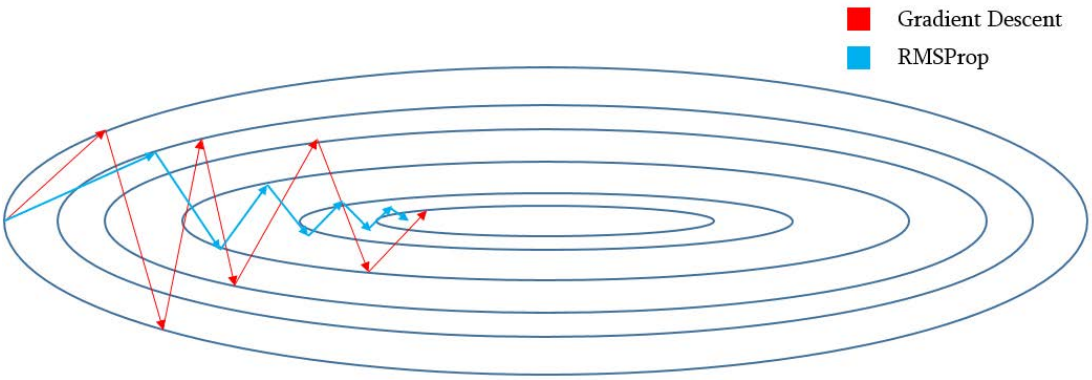


Figure 3.7: Comparison between gradient descent and RMSProp.

The equations for the update at the t-th iteration for each mini-batch are

$$\mathbf{s}_t := \beta \mathbf{s}_{t-1} + (1-\beta)\left(\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{t-1}}\right)^2 \tag{3.19}$$

$$\mathbf{W}_t := \mathbf{W}_{t-1} - \eta \frac{\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{t-1}}}{\sqrt{\mathbf{s}_t} + \epsilon} \tag{3.20}$$

where the subscript t and $t-1$ refers to the iteration number, β is a hyperparameter (usually 0.999) and ϵ is a parameter used for numerical stability (e.g. 10^{-8}). At the first iteration, $\mathbf{s}_t$ is initialized to 0.

In this variant, the weights are updating by scaling the learning rate by decaying average of the squared gradient.

RMSProp has shown good performance in practice, reducing oscillation and improving the speed of convergence.

The comparison between GD and momentum is shown in Figure 3.7.

Adam

Adam which stands for Adaptive Moment Estimation is another method that computes adaptive learning rates [21]. Adam is a combination of the previous two variants, momentum and RMSProp: it performs an exponentially decaying avarage of past squared gradients like RMSProp and exponentially decaying avarage of past gradients like momentum.

First of all, the decaying avarages, also known as exponentially weighted moving avarages,

of the past gradients and the past squared gradients are computed as follows

$$\mathbf{v}_t := \beta_1 \mathbf{v}_{t-1} + (1 - \beta_1) \frac{\partial C}{\partial \mathbf{W}_{t-1}} \tag{3.21}$$

$$\mathbf{s}_t := \beta_2 \mathbf{s}_{t-1} + (1 - \beta_2) \left(\frac{\partial C}{\partial \mathbf{W}_{t-1}} \right)^2 \tag{3.22}$$

where the subscript t and $t-1$ refers to the epoch number, β_1 and β_2 are hyperparameters (usually 0.9 and 0.999, respectively). At the first iteration, $\mathbf{v}_t$ and $\mathbf{s}_t$ are initialized to 0.

The previously computed values are biased towards zero, especially during the initial epochs. So a bias-correction ($\hat{\mathbf{v}}_t$ and $\hat{\mathbf{s}}_t$, respectively) must be calculated as follows

$$\hat{\mathbf{v}}_t := \frac{\mathbf{v}_t}{1 - \beta_1^t} \tag{3.23}$$

$$\hat{\mathbf{s}}_t := \frac{\mathbf{s}_t}{1 - \beta_2^t} \tag{3.24}$$

Finally, the update of the weights can be computed as follows

$$\mathbf{W}_t := \mathbf{W}_{t-1} - \eta \frac{\hat{\mathbf{v}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon} \tag{3.25}$$

where ϵ is a parameter used for numerical stability by avoiding division by zero (e.g. 10^{-8}). Adam algorithm is proved to be very effective in practice.

3.2.4 Overfitting and Regularization

In order to evaluate the performance of the neural network, the dataset is split into two parts: a training set and a test set. The training set is used to train the network by adjusting all the weights of the model by using optimization algorithms, like the ones discussed in the previous sections, and the performance is tested on the test set in order to understand how much the network well generalizes on samples that it has never seen before.

The errors computed on the training and test sets are called training error and test error respectively. Sometimes it happens that the training error is very small that is the network predicts good results on the training set, while the test error is large. This could appear especially when the model is very complex that in the context of the NNs means that there are too many layers or neurons in the network. The gap between the training and the test errors is

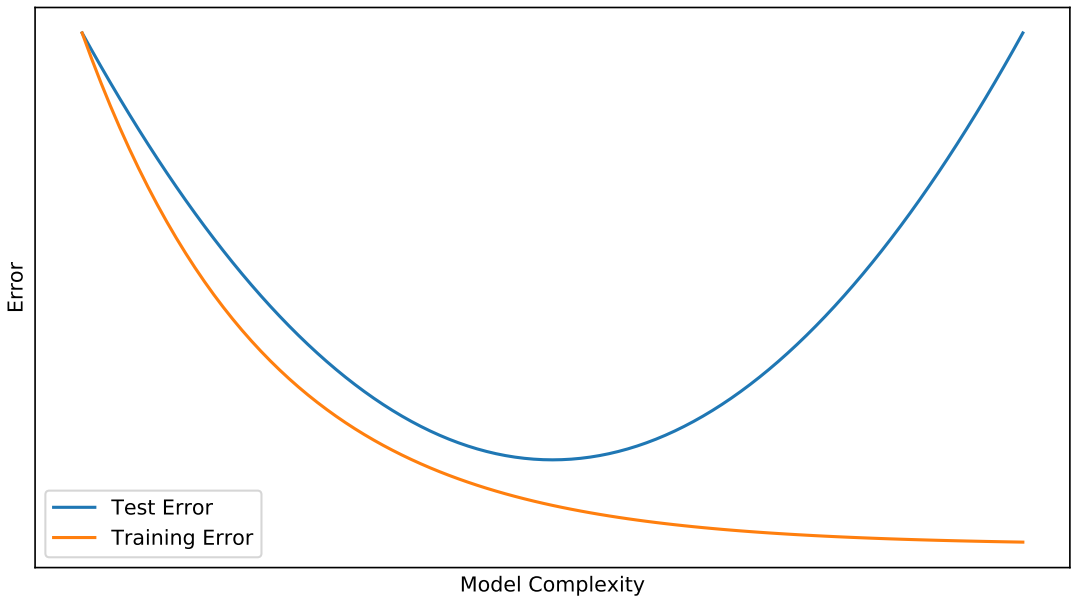


Figure 3.8: Training and test errors as a function of the model complexity.

referred to as *overfitting*. The way the training and test errors vary as a function of the model complexity is depicted in Figure 3.8.

In particular, in this context it is important to understand prediction errors and try to minimize bias and variance. The bias is the error from wrong assumptions in the learning algorithm, while the variance is an error from the variability of the training set. A model with high bias oversimplifies the model, by leading to a high error in both training and test sets. Model with high variance does not generalize on the data which has not seen before. If the model is too simple it may have high bias and low variance, while if it is too complex it might have high variance and low bias. So a good balance between these two types of error is needed.

The procedure of splitting the dataset into training and test sets is itself a way to prevent overfitting, but often this is not sufficient.

A solution which can be handy to prevent overfitting is called *regularization*, that is the imposition of constraints on the neural network. Among the ways to regularize the model, there are two techiques that are usually used for NNs: *L2-regularization* and *dropout*.

L2-regularization

L2-regularization also known as *Ridge Regression* or *weight decay* is the most common regularization technique used for NNs and machine learning models. This method consists in adding an extra term in the cost function that is the sum of the squared of all the weights, multiplied by a new hyperparameter λ which balances between the data-dependent error and the regularization term. The new cost function becomes

$$\tilde{\mathcal{L}} = \mathcal{L} + \frac{1}{2m}\lambda \sum_{w \in \mathbf{W}} w^2 \tag{3.26}$$

where $\mathcal{L}$ is the cost function such as mean squared error or cross-entropy defined previously, m is the size of the dataset and $\mathbf{W}$ is the matrix of weights of the network.

The regularization term helps the optimization algorithms to adjust the weights in a way which avoid them to become too large because large weights in neural networks are possibly a symptom of poor convergence or overfitting to the training set. A similar approach is called *L1-regularization* in which the square of the weights is replaced by the L1-distance.

Dropout

Another effective regularizaton technique especially used in deep neural networks is called dropout.

This method has been introduced by Nitish Srivastava et al. [22] in 2014. At every iteration, during the execution of the optimization algorithm, a set of neurons are selected and deactivated or dropped out along with their connections. Which neurons to be deactivated are randomly chosen at each iteration, according to a hyperparameter called *dropout probability*. For example, if this hyperparameter is set at 0.2, in average 20% of the neurons are dropped out in that iteration. This forces the network to learn a more balanced representation because it does not allow the network to be too much dependent on some neurons, thus preventing overfitting. Figure 3.9 shows how the technique works.

3.2.5 Hyperparameter Tuning and Validation

The aforementioned techniques are good to improve the performance of the optimization algorithms and to prevent overfitting, but there are crucial decisions to make before optimization begins. It is very important how the parameters and the hyperparameters of the network are chosen. The parameters of the NN include the number of layers and the num-

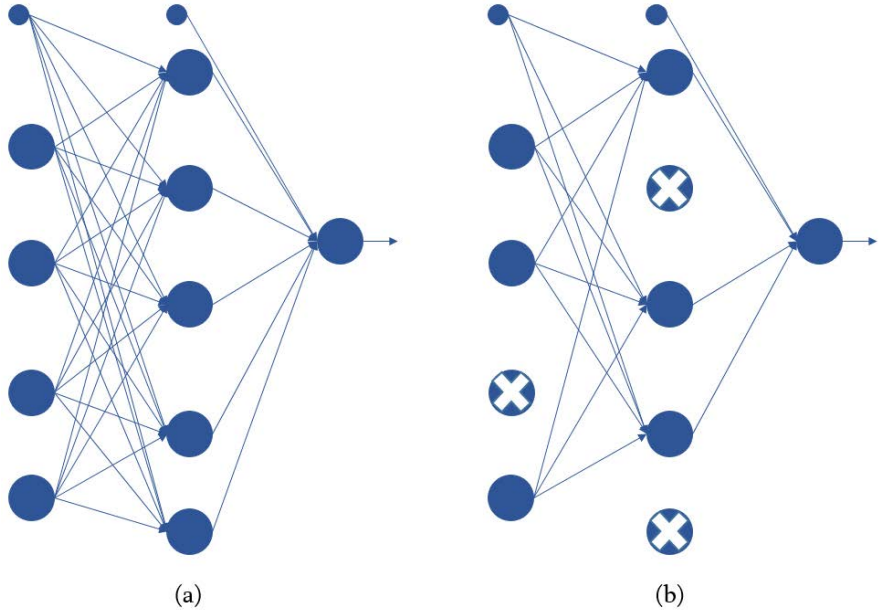


Figure 3.9: (a) Standard Neural Network and b) Neural Network after applying dropout

ber of neurons in each layer, while among the hyperparameters there is the learning rate η and the regularization parameter λ.

By trying different setting and by choosing the one which performs better on the test set (which it is constant in all the attempts) may cause overfitting. The solution to this problem consists of splitting the dataset into three parts: training set, validation set and test set.

The neural network is trained on the training set, then the performance is evaluated on the validation set in order to find the optimal hyperparameters. Finally, the configuration which performs better is evaluated on the test set.

Typically, 70% or 80% of the dataset is used for the training, while the rest is equally divided into validation and test sets, but it depends also on the amount of data available.

3.3 Neural Network Application: Transmission System Regression

In this section, an example regarding the application of NNs is described.

This example aims to show that neural networks are universal approximators, as stated by the *universal approximation theorem*. Such theorem states that the standard multilayer feedforward networks with a single hidden layer that contains a finite number of hidden neurons, and with arbitrary activation function are universal approximators in compact subsets

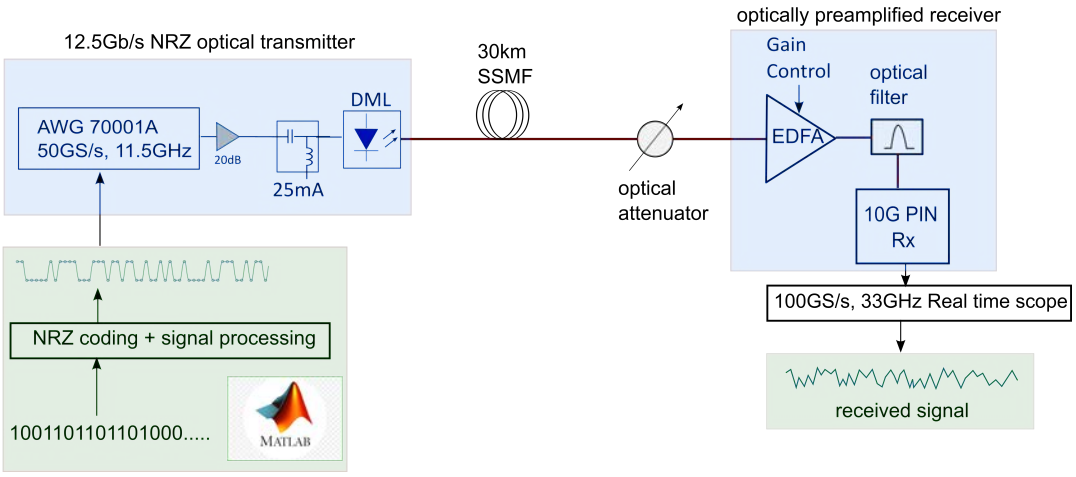


Figure 3.10: Experimental setup for the regression task. A PRBS with NRZ coding and filtered by a Bessel filter is generated through Matlab with a bitrate of 12.5 Gb/s. The waveform is sent to an AWG with 50 GS/s and is transmitted through a DML. The channel is composed of a 30 km optical fibre, followed by an optical attenuator. The receiver is composed of an amplifier, an optical filter and a 10 GHz bandwidth photodiode. The signal is recovered is using a real-time scope with sample rate of 100 GS/s.

of $\mathbb{R}^n$ [23].

Consider a DML with a bias current of 25 mA and an extinction ratio of 7 dB. Let's also fix the signal-to-noise ratio (SNR) which represents the ratio between the energy of the transmitted signal and the energy of the noise, by fixing the received power at -12 dBm and the transmission length of 30 km through an optical fibre. The objective is to build a NN which can distort an ideal waveform in the same way the whole communication system with the DML does, after the transmission through 30 km optical fibre, by fixing the DML parameters and receiving power. This is a regression task.

The experimental setup is depicted in Figure 3.10. The binary sequence is generated by a PRBS generator with a bitrate of 12.5 Gbit/s and the waveform is created through Matlab using an NRZ coding. The waveform is then smoothed by a Bessel filter and sent to an arbitrary waveform generator (AWG) with a sample rate of 50 GS/s. The AWG generates the electrical signal used by the DML to modulate the light and to send it through a 30 km optical fibre. After the fibre, there is an optical attenuator to emulate a passive splitter of a real scenario. The receiver is composed of an Erbium-Doped Fiber Amplifier (EDFA), an optical filter and a photodiode with received bandwidth of 10 GHz. The signal is finally recovered by using a real-time scope with larger received bandwidth and a sample rate of 100 GS/s.

By showing that a NN can generate the distorted received signal, given in input the ideal

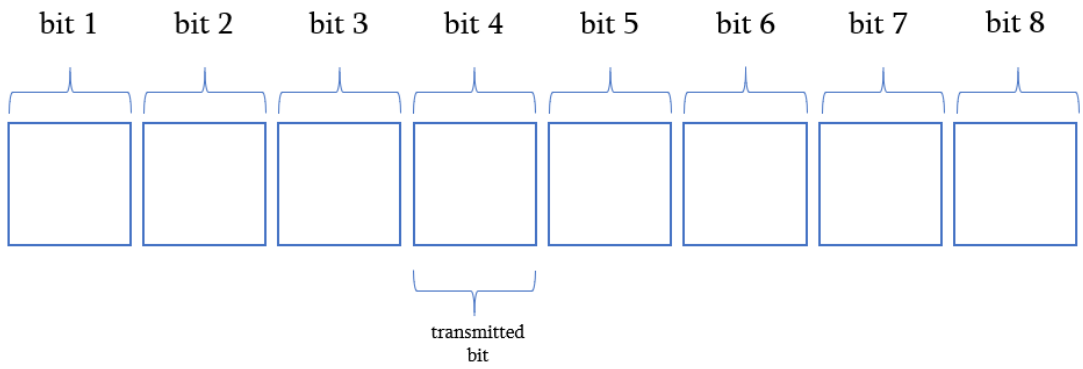


Figure 3.11: Dataset input of NN application

transmitted waveform, it may be possible to use it also for generating a predistorted signal in order to precompensate the chromatic dispersion in the optical fibre.

3.3.1 Dataset

The dataset is composed of two main parts. The training set has $99,000$ samples. Each input sample corresponds to a window of 8 taps with one sample per symbol, whose value in the middle corresponds to the sample of the transmitted bit, while the other corresponds to samples of preceding and following bits of the transmitted sequence (see Figure 3.11).

The target is one real value, corresponding to the sample of the bit, received by a photodiode with a bandwidth of 10 GHz and distorted by the whole transmission system, with the DML and the chromatic dispersion in 30 km optical fibre. The transmitted sequence is ideal, an NRZ sequence smoothed by a Bessel filter. The sequence is a PRBS16 and the signal is generated through an AWG which rescale the signal power in the range $[-1, 1]$.

The test is composed of 4 sets of $262,144$ samples each and the sequence is PRBS15, in order to avoid overfitting in the training because the neural network can learn a pseudorandom binary sequence [24].

3.3.2 NN Architecture and Training

The same dataset is used for training different neural network architectures (*NN1*, *NN2* and *NN3*), with a different number of layers, the same activation function and a different number of neurons in the hidden layers. As aforementioned, the number of neurons in the input layer is 8 and 1 neuron in the output layer with a linear activation function. The architectures of the three neural networks are summarized in Table 3.1.

	NN1	NN2	NN3
Input size	8		
Number of hidden layers	1	1	2
Neurons hidden layers	10	32	32 and 32
Activation function hidden layers	Sigmoid		
Output size	1		
Activation function output layer	Linear		
Total parameters	101	321	1377

Table 3.1: Summary of NN architectures for regression task

The training set is split into two parts. The first part is composed of $66,330$ samples (66% of data) and it is used for the training, the second one, used for validation, has $32,670$ (33%).

The three NNs are trained for 400 epochs with Adam algorithm and the cost function is the mean squared error. No regularization techniques are used because they have been proved to be ineffective.

3.3.3 Results

To evaluate the performance of the neural network, the mean square error score in the test sets is not reliable enough, because the output values to predict are in the range $[-1, 1]$ and the error is squared which leads to very small values that does not allow us to compare the performance of the different architectures. For this reason, performance is graphically evaluated.

Firstly, the shape of histograms which represent the stochastic BER is compared. Figure 3.12 shows the histogram of the actual transmission at 12.5 Gb/s with ISI = 3.

To understand when the neural network can capture the whole transmission system and generate the same distorted received signal, the neural networks, after the training, are applied to the transmitted waveform. Figure 3.13 shows the histograms of the emulated received signals after applying the trained neural networks.

As you can see from the histograms, NN1 is not capable of generating a signal as distorted as the whole transmission system does. By increasing the number of neurons in the only hidden layer from 10 to 32, we obtain a histogram shape close to the actual one, but it is still not precise enough. By adding an extra hidden layer, instead, both the shape and stochastic BER is very similar to the actual one.

Figure 3.14 shows the prediction of NN3 which is the one which emulates the whole trans-

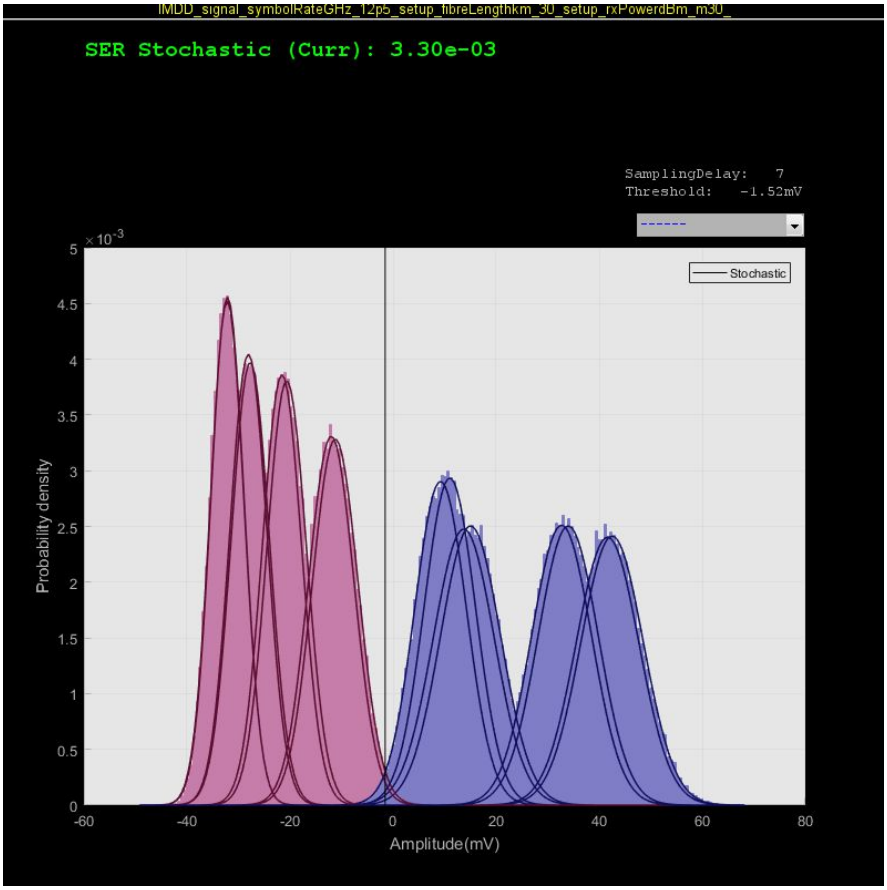


Figure 3.12: Histogram of the actual transmission with bitrate of 12.5 Gb/s and NRZ coding.

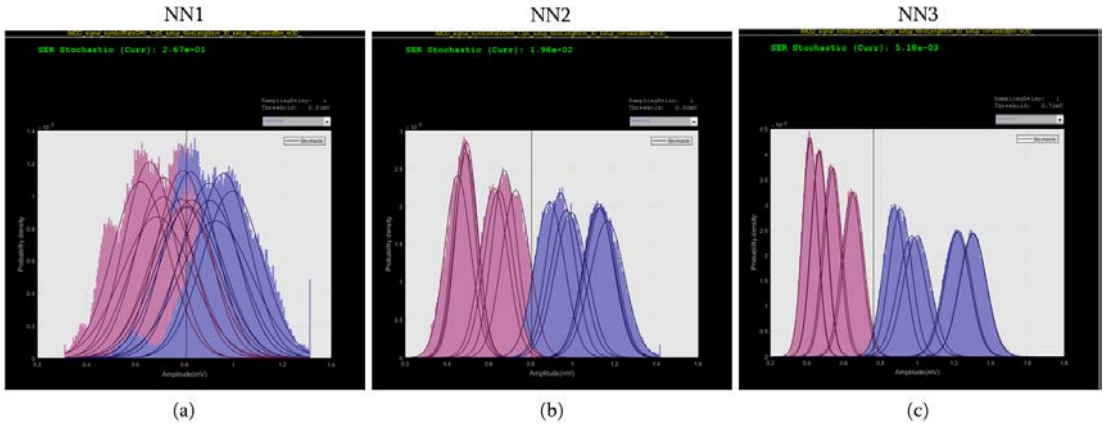


Figure 3.13: Histogram of the emulated transmission. The received signal is generated by applying the three neural network on the test set, after the training.

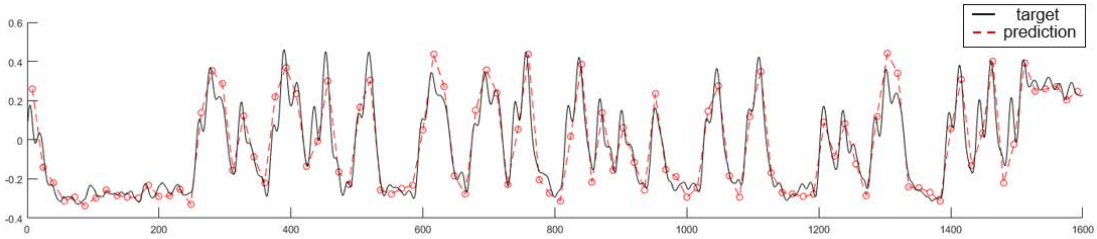


Figure 3.14: Predicitons on the first test test for NN3.

mission better, for the first 128 transmitted symbols versus the target to predict. As you can see, the received signal generated by applying the neural network is very close to the actual received signal.

This simple example shows a NN can generalize any non-linear function. This result proves that NNs as universal approximator might be a viable solution for the generation of a predistorted signal to precompensate the distortion of the chromatic dispersion with the DML because they are able also to capture the stochastic distribution of the transmitted symbols. However, according to this example and to experimental results, one more hidden layer is necessary in order to achieve good performance.

3.4 Convolutional Neural Networks

Convolutional neural networks [25] [26] are neural networks variation for processing data such as images, characterized by a grid-like topology. The name refers to the fact that, instead of using a general matrix multiplication in all layers, the convolution operation is applied. The major advantage of CNNs is training less parameters for multidimensional input data.

The continuous time convolution of x and w, generally denoted with an asterisk, is defined as

$$s(t) = (x * w)(t) = \int x(\tau)w(t-\tau)d\tau \tag{3.27}$$

while the discrete time convolution is given by

$$s(t) = (x * w)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau) \tag{3.28}$$

In the context of convolutional networks, x corresponds to the *input*, while w is referred to as the *kernel*. They are both usually a multidimensional array of data and parameters,

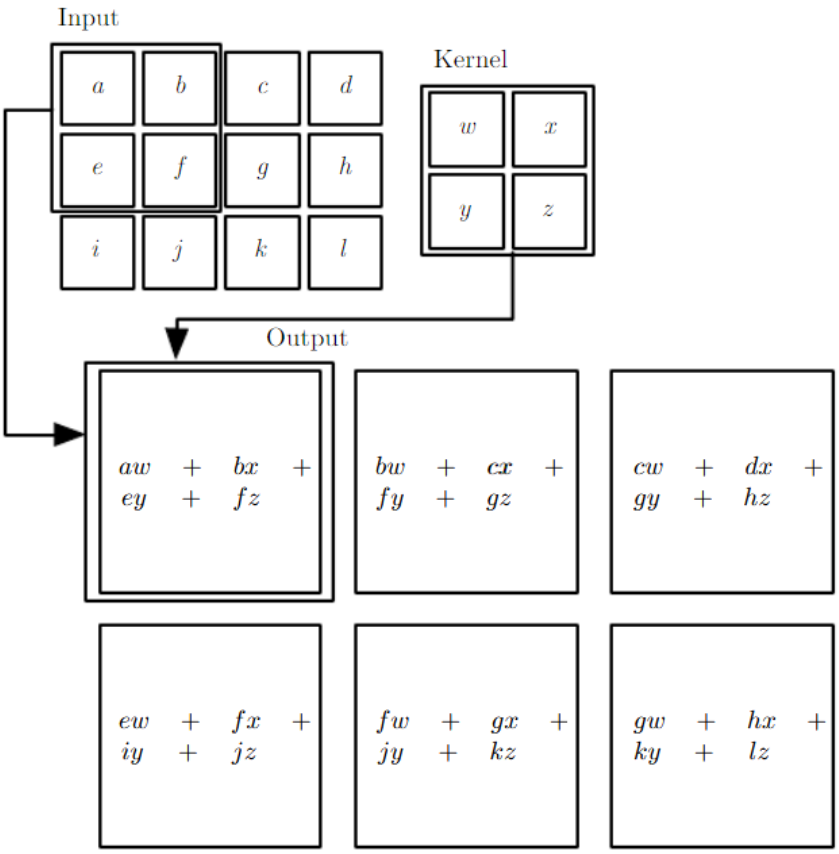


Figure 3.15: An example of 2-D convolution [27]

respectively. An example of convolution is depicted in Figure 3.15.

A hidden layer typically consists of three phases. In the first phase, several convolution operations are performed. After that, a nonlinear activation function is applied to the set of linear activations, output by the convolutional phase. Finally, a pooling function is applied. The output of a convolutional layer is called *convolved feature map*, while the ouput of a pooling layer is called *pooled feature map*.

A pooling layer reduces the size of the multidimensional array, by operating in each feature map independently and replacing the output of the net at a certain location with a summary statistic of the nearby outputs. There are two very common approaches used in pooling: *max pooling* and *average pooling*.

In the max pooling, the output is the maximum value within a rectangular neighbourhood, while in the average pooling the output is the average of the rectangular neighbour-

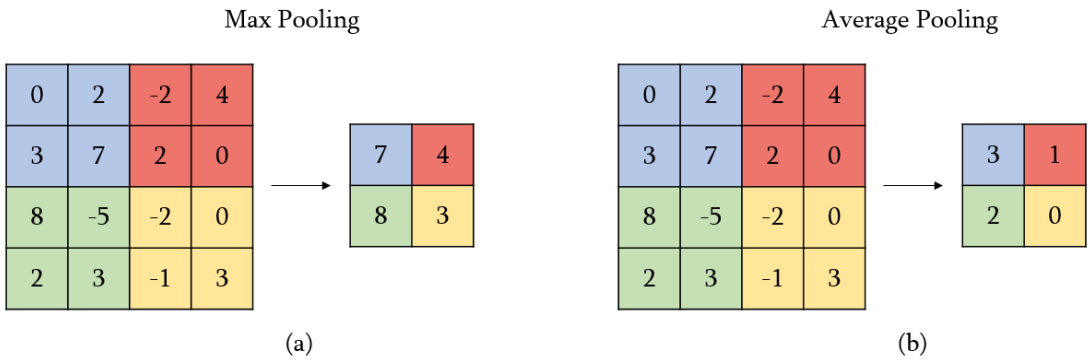


Figure 3.16: An example of (**a**) max pooling and (**b**) average pooling with filter of size 2×2

hood. The size of the rectangle depends on the size of the filter. An example of both techniques is shown in Figure 3.16.

There are two main parameters that can be set in order to change the behaviour of each layer called *stride* and *padding*.

Stride controls how the filter convolves around the input array that is how many units are shifted at a time. In the example of Figure 3.16, the stride is set at 2.

By applying a convolutional filter, the output size is smaller than the input size. For this reason, padding allows controlling the output size, by padding the input array with zeros around the border.

Let W_I be the input width, K the filter size, P the padding and S the stride. The output width W_O can be computed as follows

$$W_O = \frac{W_I - K + 2P}{S} + 1 \tag{3.29}$$

The same equation can be used to compute the output heigth H_O by replacying W_I by the input height H_I.

Finally, at the end of the CNN, there is a fully connected layer of neurons that represents the final learning phase, which maps the extracted features to desired outputs. This can be seen as a regular neural network layer in which the grid-like topology or tensor is mapped to a one-dimensional array (flatten operation). An example of CNN architecture is depicted in Figure 3.17.

CNNs have been proved to be very effective in many applications within computer vision such as face and object recognition, image classification, document analysis and in the field of speech recognition and text classification for natural language processing.

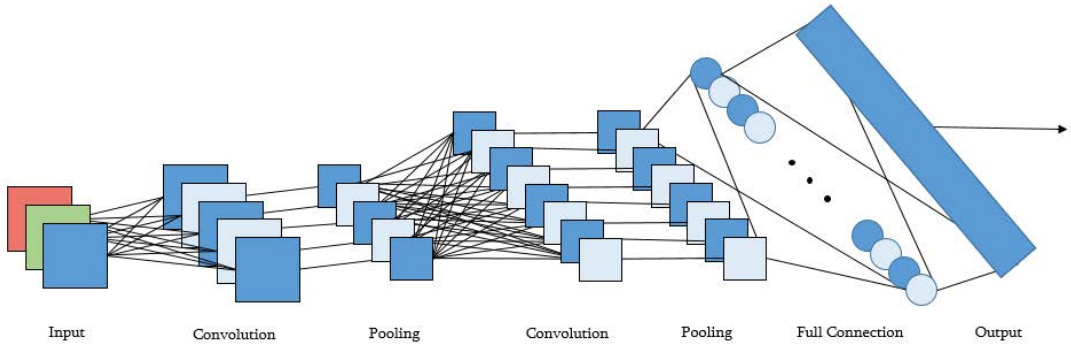


Figure 3.17: An example of CNN architecture

3.5 CNN Application: Eye Diagram Analyzer

CNNs have become the go-to method for computer vision tasks. Any data that has spatial relationships are ripe for applying CNN.

In the following example, CNN is used for detecting the amount of distortion in an optical signal by analyzing the shape of the signals eye diagram at the receiver. Given an approximate value of the amount of dispersion at the receiver, it is possible to roughly approximate the length of the transmission length. Therefore, CNN is able to extract some features from the eye diagram which allow it to predict the amount of dispersion. In this case, a multiclassification task is performed, that means the CNN will predict a range of dispersion values according to the eye diagram.

3.5.1 Dataset

The original dataset consists of 1545 jpeg images of size 490×346 pixels in RGB format, collected by a 50 GHz bandwidth oscilloscope in the laboratory. Each eye diagram has an amount of dispersion between 0 and 900 ps/nm/km with a step of 10 ps/nm/km and about 15 images for each value of dispersion, which is a time-shift of the same image. Figure 3.18 shows three images of the dataset with 0, 450 and 900 ps/nm/km dispersion, respectively.

Because the actual size of the images is too large and it would lead to a very slow training, the images are reshaped to a size of 28×28 and 64×64 pixels and they constitute two different datasets, whose performance are compared. Furthermore, the RGB images are converted to grayscale. Because a pixel of a grayscale image has a value between 0 and 255, these values are normalized in the range $[0, 1]$ by dividing each pixel by 255 in order to achieve faster training.

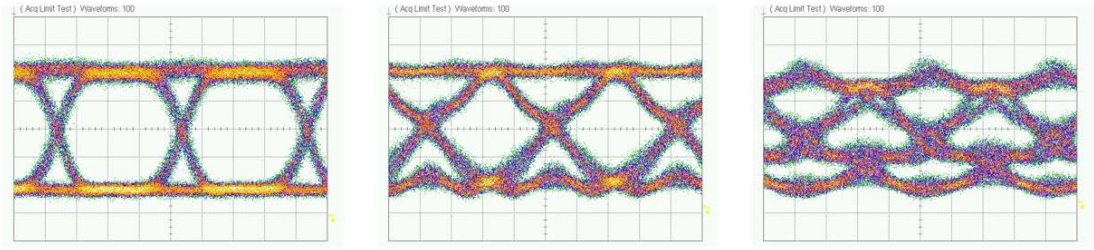


Figure 3.18: Eye diagrams captured with a 50 GHz bandwidth oscilloscope with dispersion of $(\mathbf{a})$ 0 ps/nm/km, $(\mathbf{b})$ 450 ps/nm/km and $(\mathbf{c})$ 900 ps/nm/km.

To each image, a label in the range $0-8$ is assigned. The label 0 corresponds to a value of dispersion between 0 and 90 ps/nm/km, the label 1 corresponds to a dispersion between 100 and 190, up to the label 8 which corresponds to a dispersion between 800 and 900 ps/nm/km.

The dataset is split into two parts: 1236 images (80%) for the training and 309 images (20%) for the test set. The training set is further split into 927 images (75%) for the proper training, while the remaining 309 images (25%) is used as a validation set. A similar number of representatives for each class in each set is guaranteed (stratified sampling).

3.5.2 CNN Architecture and Training

The CNN architecture is the same for both the dataset of images of size 28×28 and 64×64 pixels. There are three convolutional layers whose kernel size is 3×3 alternates with of two max pooling layers whose filter size 2×2. The third convolutional layer is followed by a flatten operation and a dense layer with 128 neurons. The activation function (where applicable) is ReLU function. The output layer is a softmax layer with 9 neurons, one for each possible outcome. The output shape of each layer and the number of parameters that must be trained are summarized in Tables 3.2 and 3.3. No regularization operations are applied because they have been proved to be ineffective.

The CNN is trained for 35 epochs for images of size 28×28 to prevent overfitting and 100 for images of size 64×64.

The cost function is the categorical cross entropy and the error is backpropagated using Adam algorithm.

Layer	Filter shape	Activation	Output shape	Number of parameters
Convolutional	$(3, 3)$	ReLU	$(26, 26, 16)$	160
Max pooling	$(2, 2)$	NA	$(13, 13, 16)$	0
Convolutional	$(3, 3)$	ReLU	$(11, 11, 16)$	$2,320$
Max pooling	$(2, 2)$	NA	$(5, 5, 16)$	0
Convolutional	$(3, 3)$	ReLU	$(3, 3, 16)$	$2,320$
Flatten	NA	NA	144	0
Dense	NA	ReLU	128	$1,8650$
Dense	NA	Softmax	9	$1,161$
Total parameters: $24,521$				

Table 3.2: Summary of CNN architecture for 28×28 images

Layer	Filter shape	Activation	Output shape	Number of parameters
Convolutional	$(3, 3)$	ReLU	$(62, 62, 16)$	160
Max pooling	$(2, 2)$	NA	$(31, 31, 16)$	0
Convolutional	$(3, 3)$	ReLU	$(29, 29, 16)$	$2,320$
Max pooling	$(2, 2)$	NA	$(14, 14, 16)$	0
Convolutional	$(3, 3)$	ReLU	$(12, 12, 16)$	$2,320$
Flatten	NA	NA	2304	0
Dense	NA	ReLU	128	$295,040$
Dense	NA	Softmax	9	$1,161$
Total parameters: $301,001$				

Table 3.3: Summary of CNN architecture for 64×64 images

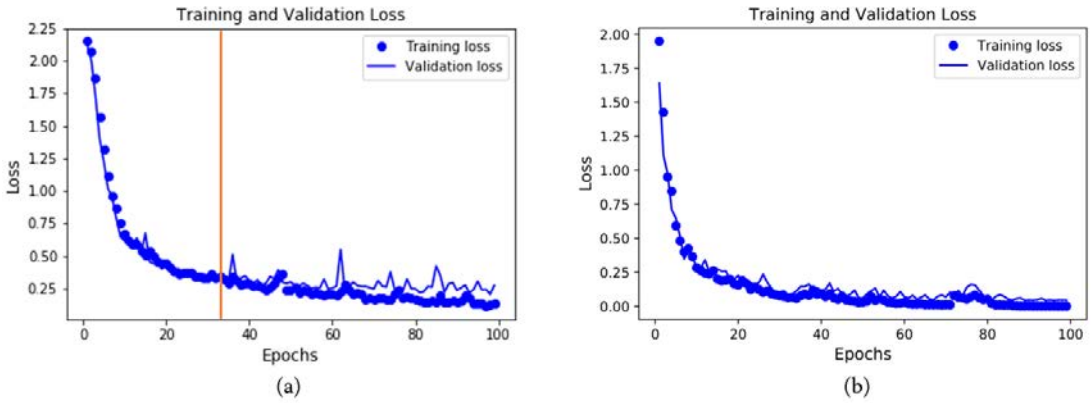


Figure 3.19: Training and Validation loss for images of size (**a**) 28×28 and (**b**) 64×64 pixels.

3.5.3 Results

The total number of parameters to train in the CNN with input images of size 64×64 is more than twelve times the parameters for the training on 28×28 pixels images. For this reason, the number of epochs required for the training in the first case is higher. Figure 3.19 shows the training and validation loss of the aforementioned datasets.

As you can see, the training on the dataset with images of size 28×28 pixels starts slows because the curve has a flat area at the beginning, while after a few epochs there is a steep slope and the loss decreases quickly for both the training and the validation set. The backpropagation algorithm is stopped after 35 epochs because of a divergence between the training and the validation loss, symptom of overfitting. For the second dataset, instead, CNN starts learning faster by reaching a loss very close to zero after around a few epochs. Because the training and validation loss is similar for many other epochs, the backpropagation algorithm is applied until the 100^{th} epoch.

The accuracy achieved by the CNN on the training, validation and test sets are summarized in Table 3.4. In the first dataset with images of size 28×28, the CNN reaches about 90% accuracy in the three sets, with 34 errors out of 309 in the test. For the second dataset with image size 64×64, the training and validation accuracy is close to 100% and an accuracy of 96% in the test, with only 12 errors in total. This means that by giving higher resolution images as input, the CNN is able to capture in an easier way some features required for understanding the amount of dispersion.

This simple example shows the great power of CNNs in image processing and their potential applications in photonics. For example, a pretrained CNN can be part of the oscilloscope

Images size	28×28	64×64
Number of epochs	35	100
Training set accuracy	91.48%	100%
Validition set accuracy	90.29%	98.71%
Test set accuracy	89.32%	96.12%

Table 3.4: Results of the training and accuracy on the training, validation and test sets.

in order to carry out further processing for analyzing the performance of the transmission.

4

Particle Swarm Optimization

While the behaviour of SGD for convex cost function (or objective function) is deeply understood and works well in practice, when the objective function is not convex this algorithm starts struggling. Moreover, in the context of predistorted signals that compensate the system impairments in DML, the target is unknown and the NNs must operate in the context of unsupervised learning. For these reasons, the training of the model cannot be done conventionally and requires indirect strategies, for example, techniques that involve a fitness function that captures the overall system performance (such as the BER of the received signal) to be minimized respect to the NN parameters. Particle swarm optimization (PSO) algorithm, that will be described in the next sections, could be suitable for solving this problem.

4.1 PSO Algorithm

Particle swarm optimization algorithm is a nature-inspired population-based metaheuristic algorithm, originally developed by Eberhart, Kennedy, and Shi [28]. A metaheuristic is a partial search algorithm whose aim is to find enough good solution to an optimization problem [29]. PSO is population-based because instead of improving one single solution iteration by iteration, it maintains a set of candidate solutions and iteratively tries to improve all of them. Moreover, it is in the class of nature-inspired metaheuristics, which are inspired by natural systems. In particular, PSO mimics the social behaviour of birds flocking and fishes schooling. This situation can be seen in everyday life. Someone starts feeding some fishes in an

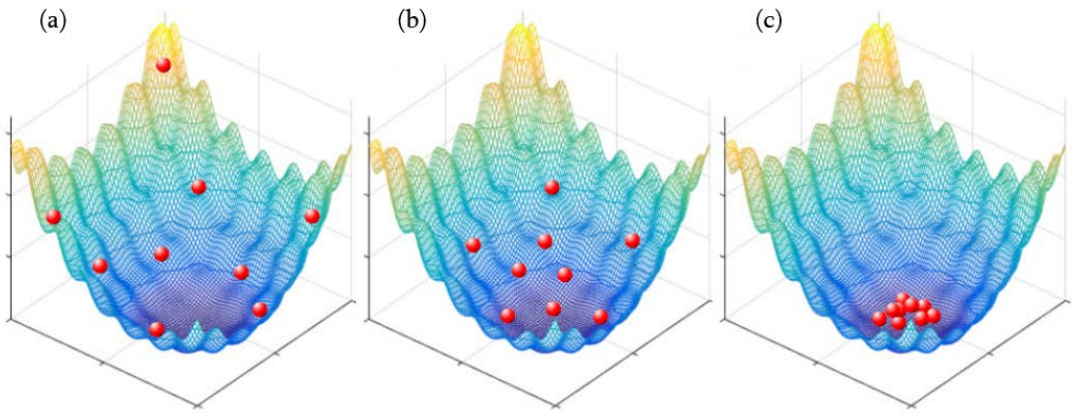


Figure 4.1: Particle Swarm Optimization evolution in following iterations in a 2-dimensional space. The red particles are searching for the minimum of the surface [30].

aquarium and the fish closer to the fish food starts eating. After a while, a second fish comes, and it does not take much time until a small school is close to the water surface where there is fish food. From a mathematical point of view, like a global optimization algorithm should do, the fish school found the global optimum of the objective function that is the area of the water surface plenty of food.

An example of the evolution of the algorithm is shown in Figure 4.1. The particles are moving in a 2-dimensional space and they aim to find the minimum of the objective function.

Starting from a randomly initialized set of particles, that represents a set of potential solutions, the algorithm tries to improve those solutions according to a quality measure called *fitness function*. The solutions get improved by moving the particles around the multidimensional search space thanks to a set of mathematical expressions that model some interparticle communications. The initial idea was to generate a set of multidimensional points, whose values represent their position in the search space and an initial velocity vector for each. Using those velocity vectors, the particles were able to change their position iteratively, while the velocity was adjusted randomly. One of the first application was applying the PSO algorithm to model fuzzy objects [31]. Later on, the concept of interparticle communication has been introduced, with which the particles were able to change their position according to the best solution found by the entire swarm [32].

In the next section, a more detailed explanation of how the algorithm works is provided.

4.1.1 Hyperparameters and Update Rule

In the simplest version of the algorithm, there are five important hyperparameters that must be tuned, depending on the application in which the PSO algorithm is applied.

- Number of particles in the swarm: depending on how much complex is the fitness function to optimize, the number of particles in the swarm varies. Usually, the harder is the problem, the greater the number of particles must be. Moreover, by initializing the algorithm with a high number of particles, is more likely that the algorithm converges faster. The drawback is in the computational cost of the algorithm that increases.

- Number of iterations: this parameter fixes the number of updates of the position of all the particles in the swarm. However, a predetermined number of iterations is not a good criterion that guarantees to obtain enough good solution to the problem. Because of that, usually, other termination criterions are set, such as running the algorithm until a good solution is found or stopping the execution when there are no improvements on the quality of the solution for a certain number of iterations.

Furthermore, there are other three hyperparameters that influence the convergence of the algorithm and the update of the positions of the particles in each iteration.

- Inertia weight w

- Cognitive costant c_1

- Social constant c_2

Before explaining the aim of these three hyperparameters, the update rule is introduced. To do so, first, let's introduce some notation

- $\mathbf{x}_i^k$: position of the particle i at iteration k

- $\mathbf{v}_i^k$: velocity of the particle i at iteration k

- $\mathbf{xlbest}_i^k$: best position of particle i at iteration k

- $\mathbf{xgbest}^k$: best position of the whole swarm at iteration k

- $r_{1,i}^k$ and $r_{2,i}^k$: random numbers from an uniform distribution in range $[0, 1]$, associated to the particle i at iteration k

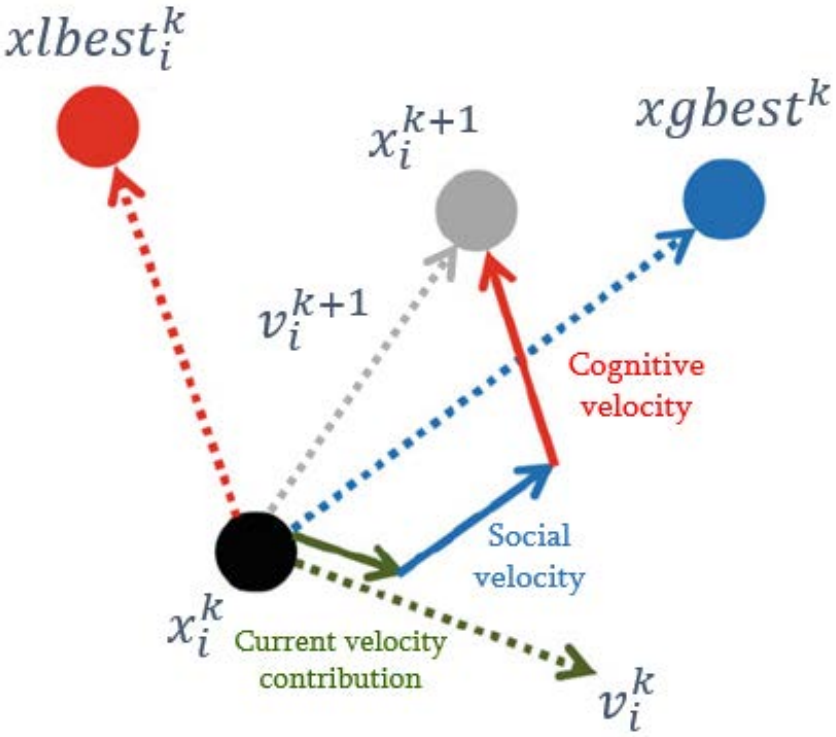


Figure 4.2: Schematic movement of the particle i at iteration k base on Equation 4.1.

The velocity $\mathbf{v}_i^{k+1}$ of the particle i at iteration $k+1$ is given by

$$\mathbf{v}_i^{k+1} = \underbrace{w \cdot \mathbf{v}_i^k}_{\substack{\text{Current velocity} \\ \text{contribution}}} + \underbrace{r_{1,i}^k \cdot c_1 \cdot (\mathbf{xlbest}_i^k - \mathbf{x}_i^k)}_{\text{Cognitive velocity}} + \underbrace{r_{2,i}^k \cdot c_2 \cdot (\mathbf{xgbest}^k - \mathbf{x}_i^k)}_{\text{Social velocity}} \tag{4.1}$$

While the position $\mathbf{x}_i^{k+1}$ of the particle i at iteration $k+1$ is given by

$$\mathbf{v}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1} \tag{4.2}$$

A schematic movement of the particle i at iteration k is shown in Figure 4.2.

The particles are able to remember the best position found up to the current iteration by themselves and by the whole particle and these two positions influence the following ones. In particular, the two constants c_1 and c_2 have an important effect. A relatively high value of c_1 will force the particles to move around their local best experiences, while higher values

of c_2 will result in faster convergence to the global best position and this is why the two constants are called *cognitive* and *social* constant, respectively. For this reason, the two constants are usually set to equal values in order to balance the influence of the local and global best solutions found up to the current iteration. The first term of the Equation 4.1, instead, is a contribution on the update based on the current velocity, according to a proportional constant called *inertia weight*, which often has a value in the range $(0, 1)$. By eliminating the first term the particles are not able to leave the small portion of the search space when they are initialized, while without any inertia weights, the particles will not converge to the known good positions. Though, the inertia weight introduces a balance between these two opposite effects. In some variants of the PSO algorithm, the inertia weight is a linear or non-linear function of the time in order to improve the convergence speed and to limit the search space in the last iterations, by decreasing its value with time.

During the update, other constraints can be added to have a higher control on the particles. For examples, if the search space is limited, the position of the particles can be bounded, or one can limit the maximum allowable velocity to avoid the particles to move too far away. In this first case, the mirror effect on the velocity is sometimes applied which the velocity is set to the opposite direction in order to force the particles to remain the search space.

The pseudocode of the PSO algorithm is listed below

Algorithm 4.1 Particle Swarm Optimization

```
Initialize x_i, v_i and xlbest_i for each particle
for each particle i
    Evaluate objective function
    Update xlbest_i and xgbest

while not termination condition
    for each particle i
        Update v_i according to Equation 4.1
        Update x_i according to Equation 4.2
        Evaluate objective function
        Update xlbest_i and xgbest
```

However, all these hyperparameters must be tuned depending on the application and on the relative ruggedness and smoothness of the hyperspace. The literature suggests some rules of thumb to accomplish this task. For instance, if the hyperspace is small a low inertia weight is more suitable, while in the case of high inertia weight, the maximum allowable velocity of

each particle should be limited [32].

One advantage of this technique is that it is very fast in term of convergence. On the other hand, fast convergence causes the algorithm to not improve the solution in further iterations. For this reason, many variants have been proposed like democratic PSO or adaptive methods [32].

4.1.2 PSO Applied to Neural Networks

In the context of machine learning, neural networks can be trained also by using metaheuristic algorithms such as PSO, by achieving enough good solutions to the optimization problem. In this case, the PSO algorithm can be slightly modified to be applied to NNs. To accomplish it, it is necessary to specify how the position, velocity and best solution are represented in the context of NNs. The position of a NN in the PSO algorithm corresponds to its set of weights that are updated iteration by iteration, causing a change in the output of the network. As when SGD is applied, the NN is randomly initialized, accordingly to a certain distribution, if required. The velocity of a NN is a set of arrays of the same dimension of its weights (or position) in order to allow the vector sum operations in the update step. Lastly, a solution of NNs in the context of PSO corresponds to the current set of weights of the network. In particular, the best solution of the particle is represented by the set of weights of the network that achieved the best fitness function value up to the iteration taken into account. Therefore, all the addends in Equations 4.1 and 4.2 have the same dimension and the update rule can be applied.

In the next section, a comparison between the PSO and the standard technique used to train a neural network is presented.

4.2 Comparison of PSO and SGD

For training a neural network, the most common technique is back-propagation, presented with many variants of the algorithm in the previous chapter. This algorithm works very well in practice for convex function and low dimensional space, but as stated in the previous section, metaheuristic algorithms can be used to train a NN. Unlike GD, PSO does not require the cost function to be differentiable and the approach is not as complicated as the use of the gradient.

In order to compare the performance on the training of a NN, the PSO is applied for a problem at the receiver in an optical communication system called *equalization*. This prob-

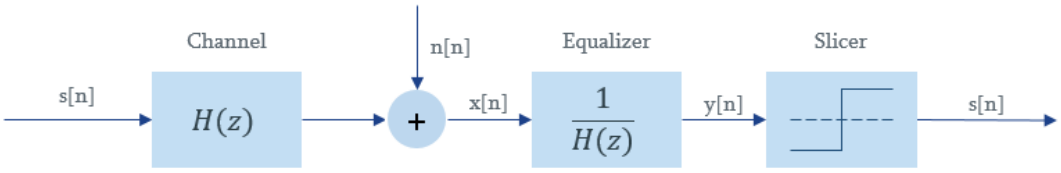


Figure 4.3: Representation of a model with linear equalizer.

lem will be briefly introduced below and then the comparison of the results of the two training techniques are discussed.

4.2.1 Equalization

The equalization deals with distortion of a transmitted signal through a channel, in particular, due to effect of ISI. The basic idea of a linear equalizer is to filter the received signal through a filter that approximates the inverse of the transfer function of the channel. Such an equalizer is often called a feed-forward equalizer (FFE) and it is implemented as a finite impulse response (FIR) filter.

The system representation is depicted in Figure 4.3. $s[n]$ represents the transmitted sequence of symbols sent through a channel whose frequency transfer function is $H(z)$. $n[n]$ is additive white Gaussian noise (AWGN), $x[n]$ is the input of the linear equalizer with frequency transfer function as close as possible to the inverse of the channel one and $y[n]$ is the output of the equalizer. The latter goes through a slicer with a certain decision threshold in order to recover the original sequence.

The FIR filter is given by

$$y[n] = \sum_{k=-N}^{N} c_k x[n-k] \tag{4.3}$$

where c_k are the $M = 2N + 1$ coefficients, knows as taps. These taps are adjusted to approximate the channel inverse, according to a metric or cost function called mean squared error (MSE), given by

$$\text{MSE} = \mathbb{E}\left[(s[n] - y[n])^2\right] \tag{4.4}$$

where $\mathbb{E}$ is the statistical expectation.

A very important aspect of chromatic dispersion from an equalization point of view is

that while chromatic dispersion in itself is linear and does not depend on the intensity of the light, its effect causes the fibre optic channel to be non-linear. This means that the linear equalizer cannot model non-linear effects, especially if the ISI of the channel is severe. For this reason, FFE is usually used with a decision feedback equalizer (DFE) which uses past symbols decisions to remove ISI from the following symbols.

However, the non-linear effects of the channel may be counteract using a NN at the receiver as a non-linear equalizer. In particular, next section will show that NNs for equalization at the receiver are promising and have better performance than FFE. Moreover, a comparison between the training of the neural network by using backpropagation and PSO is provided.

4.2.2 Dataset and Training

The dataset is generated using a simulation testbed with a MZM for the transmission. The modulation format is NRZ at 25 Gb/s bitrate over 100 km fibre. The transmitter and the receiver bandwidths at 3 dB are 12.5 GHz. The transmission sequence for the training is generated from PRBS16 with receiving power of -12 dBm.

The dataset is composed of two main parts. The training set has $99,000$ samples. Each input sample corresponds to a window of 8 taps with one sample per symbol, whose value in the middle corresponds to the sample of transmitted bit, while the other corresponds to samples of preceding and following bits of the transmitted sequence.

The target is one real value, corresponding to the transmitted bit which can be either a 0 or a 1.

The test is composed of 64 sets of $262,144$ samples each, grouped in sets of 4 for different received power in the range $[-17, -9]$ dBm with a step of 0.5 dBm. The sequence for the test is PRBS15, in order to avoid overfitting in the training.

The dataset is used for training a FFE with 8 taps and two neural networks (*NN - Adam* and *NN - PSO*) with the same architecture.

NN - Adam is a 2-layers NN with 8 neurons in the input layers, 8 in the hidden layer and 2 output neurons. In the hidden and output layers the activation function is the sigmoid function. The network is trained using Adam algorithm for 400 epochs and mean squared error as cost function. The architecture and the parameters for the training are summarized in Table 4.1.

NN - PSO has the same architecture (number of layers, number of neurons, activation functions) of NN - Adam. It is trained using PSO algorithm with mean squared error as

Parameter	Value
Input layer	8 neurons
Hidden layer	8 neurons, sigmoid activation function
Output layer	8 neurons, sigmoid activation function
Optimizer	Adam algorithm
Loss function	Mean squared error
Number of epochs	400

Table 4.1: Summary of NN - Adam parameters

Parameter	Value
Input layer	8 neurons
Hidden layer	8 neurons, sigmoid activation function
Output layer	8 neurons, sigmoid activation function
Fitness function	Mean squared error
Number of iterations	500
Number of particles	200
Inertia weight	0.3
Social constant	1.2
Cognitive constant	1.2

Table 4.2: Summary of NN - PSO parameters

fitness function. The network is trained for 500 iterations with 200 particles, inertia weight at 0.3, cognitive and social constants at 1.2. The architecture and the parameters for the training are summarized in Table 4.2.

4.2.3 Results

Figure 4.4 shows the eye diagram and the histogram after the transmission over 100 km fibre without equalization. The eye in the picture appears very distorted. For this reason, equalization at the receiver is required.

The BER over received power without equalization and with equalization is shown in Figure 4.5. The equalization is carried out by using an FFE, NN - Adam and NN - PSO.

As you can see, the use of FFE is not very effective. The performance are slightly better than not using any equalization techniques. This happens because the number of taps is not enough large when the transmission is very stressed.

The two neural networks perform much better than FFE, with an improvement of three orders of magnitude for the highest levels of received power. In particular, NN - Adam per-

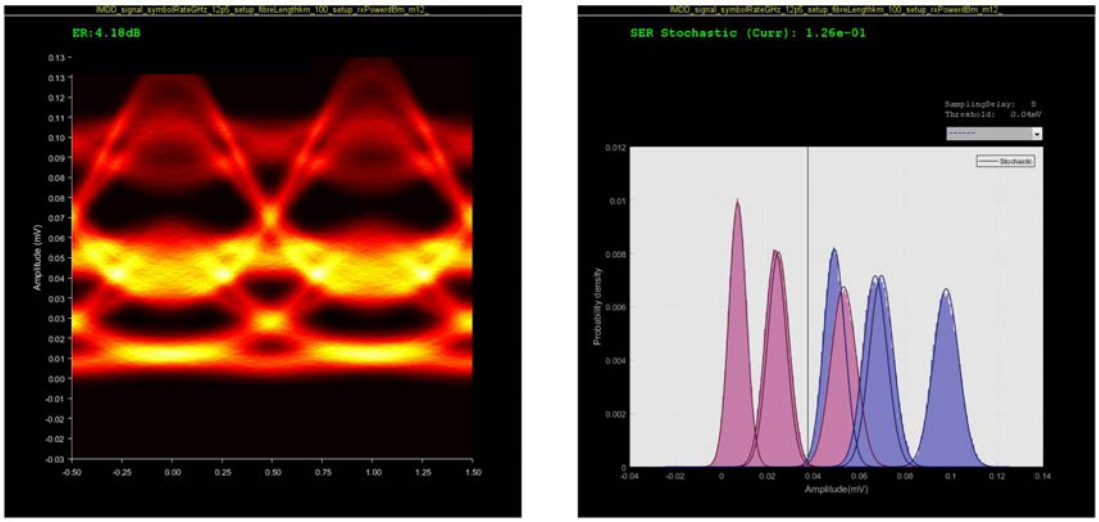


Figure 4.4: Eye diagram and histogram after 100 km optical fibre without equalization. The transmission with MZM is at 25 Gb/s with NRZ modulation.

forms slightly better than NN - PSO.

Despite the fact the NNs achieve similar results, under a computation point of view, training the neural network by using traditional techniques is more effective. This is due to the fact that the weights of the NNs vary in a wide range (about $[-20, 20]$) and hyperspace is very huge, so a random initialization can cause the particles to be very far from the optimal solution. For this reason, many attempts have been tried before finding a solution with similar performance. Moreover, because the cost function is convex, the gradient descent technique is more suitable for this application.

Another observation on the final configuration of the weights is that, at the end of the training, the weights which connect the hidden layer to the output layer are symmetric respect to 0. This might be used as an additional constraint on the PSO update in order to speed up the convergence.

However, this example shows that PSO can be a viable solution for the optimization of a NN and it demonstrates the potential of NNs at the receiver for improving the performance of the transmission over high transmission lengths.

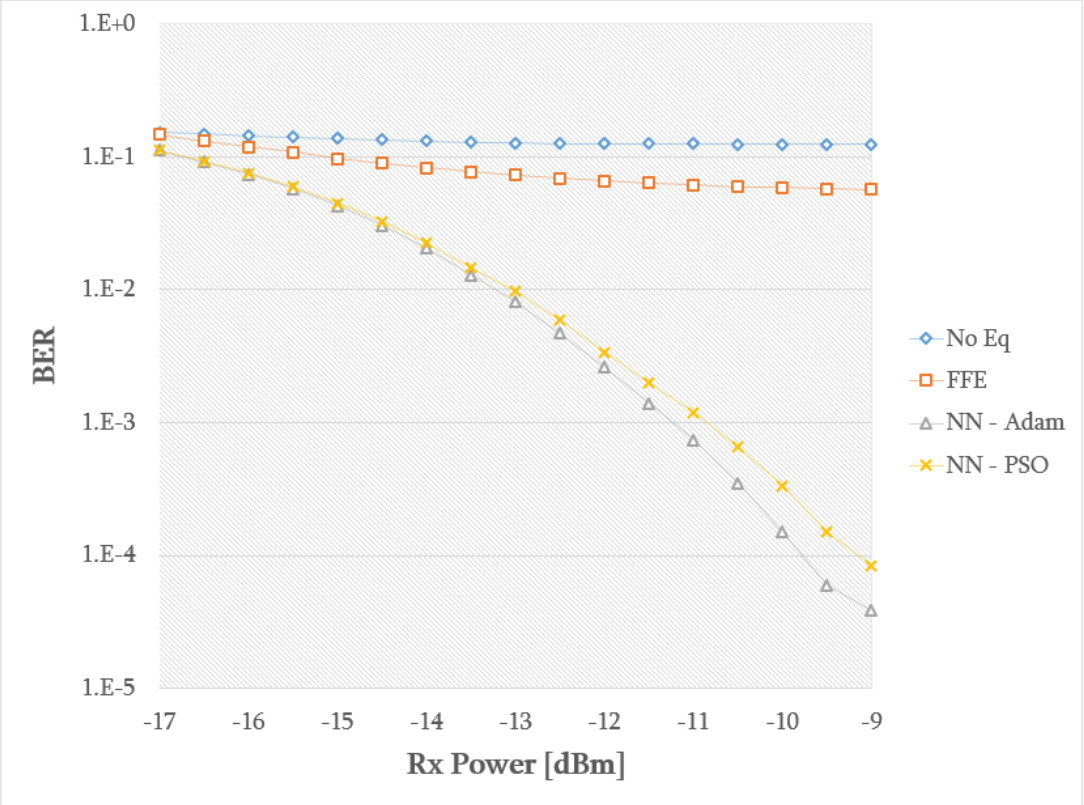


Figure 4.5: Comparison of the perfomance in terms of BER over received power without equalization and with equalization carried out by FFE, a neural network trained with Adam algorithm and a neural network trained with PSO algorithm. The equalization takes into account 8 taps on a transmission with MZM at 25 Gb/s over 100 km optical fibre with NRZ modulation.

5 Predistortion Techniques

Non-linear compensation techniques are very important to improve the performance of telecommunication channels by precompensating channel non-linearities. In this field, two categories can be distinguished: non-linear equalizers at the receiver, briefly described in the previous chapters, and predistorters at the transmitter.

In this chapter, two techniques that exploit the use of neural networks are described with results achieved in simulation whose performance seems promising for experimental applications.

5.1 Indirect Learning

As mentioned before, it is difficult to obtain the desired output when neural networks are trained with conventional methods like backpropagation algorithm. One possible solution to this problem is to use an *indirect learning architecture* to design a predistorter for precompensating the non-linearities when a DML is used to modulate the light.

5.1.1 Motivation

Indirect learning architecture approach is one the most commonly used technique for the identification of digital predistorters.

The schematic diagram of the most general indirect learning architecture is shown in Figure 5.1. This approach exploits two copies of the same model both at the transmitter and

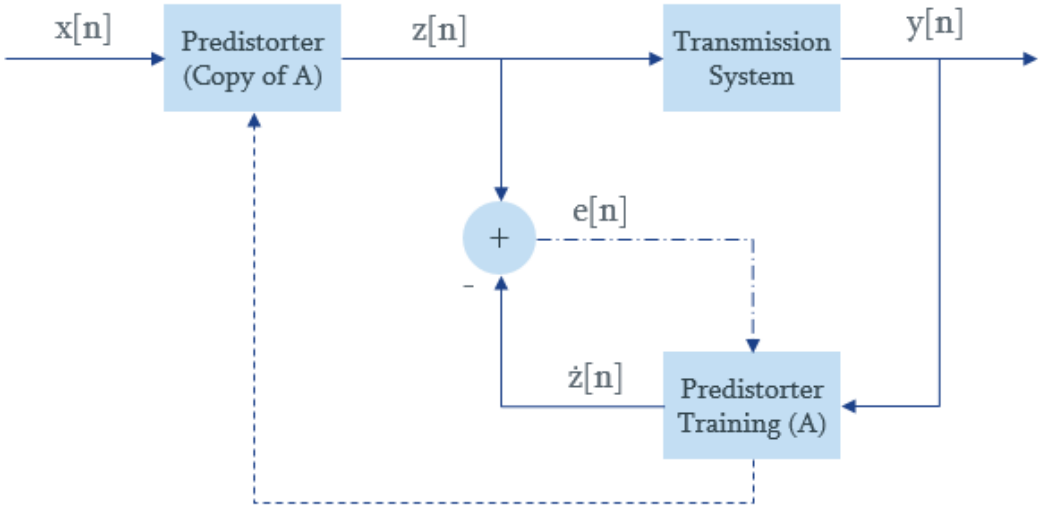


Figure 5.1: Schematic diagram of indirect learning architecture to obtain the desired output for training the neural network predistorter.

at the receiver. The input of the first model is the signal to transmit, while the input of the second is the received signal. The output of the first model is compared to the output of the second in order to compute the error and to use it for adjusting the parameters of the model for the following iterations. The procedure is repeated until the output converge, that is the error is close to 0.

It found many successful applications in compensating non-linearities channels in which the desired output is not known in advance.

For example, ILA has been presented by Eun *et al.* [33] for the predistortion on a satellite communication channel and for dealing with non-linear power amplifiers [34]. Both the applications use a p-th order polynomial based on the Volterra series model [35].

Some neural networks based indirect learning architecture approaches have been proposed to deal with non-linearities in digital communication channels [36], where most of the signals used are complex and for which a complex neural network which was formalized by Benvenuto *et al.* [37] has been used.

Based on the results on the indirect learning architecture in these applications, this approach has been used with neural networks as predistorter for DML.

5.1.2 Implementation Details

According to the model in Figure 5.1, there are two copies of the same neural network at the transmitter and at the receiver. The neural network at the transmitter is used as predistorter to obtain a signal $z[n]$ from the transmitted signal $x[n]$. $z[n]$ is then sent to an AWG for modulating it through the DML and transmitting it through the optical fibre. After the signal is detected by the photodiode at the receiver ($y[n]$), it is process by the same neural network whose output is $\dot{z}[n]$. Now, the error $e[n]$ is computed as

$$e[n] = (z[n] - \dot{z}[n])^2 \tag{5.1}$$

which is the MSE between the output of the first neural network (predistorter) and the output of the same copy, after the transmission.

Therefore, the error is backpropagated to adjust the weight of the neural network at the receiver and, at the end of the update, the weights are copied on the network at the transmitter. As the error approaches the optimum value, the received signal $y[n]$ approaches the transmitted signal $x[n]$. This procedure is repeated until the error is very close to 0. After the training, the network A is removed from the system and the predistorter works alone for precompensating the non-linearities of the system.

If the inverse of the whole transmission system does not exist, there is no guarantee on the convergence of the training phase.

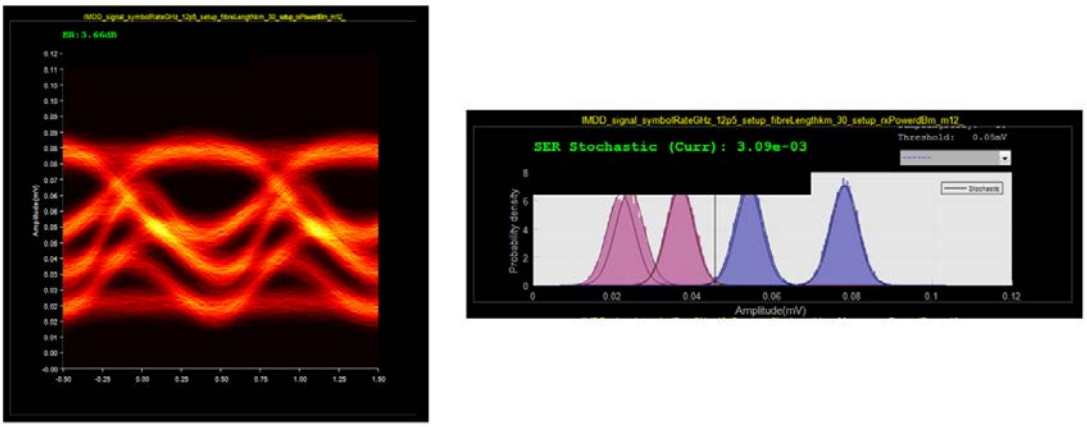
The input of the neural networks is a sequence of 10 taps with 2 sample per symbol, with 20 input values in total, and the output is a raw value corresponding to the transmitted bit which is in the middle, as seen before in Section 3.3.

The NN architecture is a 3-layers network with 32 neurons and sigmoid activation function in each hidden layer (NN_3 of the example in Section 3.3). The error is backpropagated using Adam algorithm.

The whole simulation is run in Matlab, except for the backpropagation of the error to adjust the weights which is run in Python using Tensorflow library.

5.1.3 Results

The simulation has been run for different neural network architectures, but enough good results are obtained by running 3-layers network described in the previous section. This predistortion scheme have been tried for different transmission lengths, leading to similar results, with more necessary iterations when the transmission length increases.



(a) (b)

Figure 5.2: $(\mathbf{a})$ Eye diagram and $(\mathbf{b})$ histogram without optimization. The transmission length is 30 km, bitrate at 12.5 Gb/s and -12 dBm received power.

The results of the predistortion scheme are depicted by showing the evolution of the algorithm in following iterations.

Firstly, Figure 5.2 shows the eye diagram and the histogram of the received signal with the following setup: 30 km transmission length, bitrate of 12.5 Gb/s and -12 dBm received power.

The indirect learning algorithm have been run for 20 iterations and it has been stopped when the BER was enough low. As aforementioned, at the neural network is randomly initialized. For this reason, the predistorted signal in the first iterations leads to a poor BER.

Figure 5.3 shows the evolution of the eye diagrams at the iteration 1, 5, 10 and 15. As you can see, by starting with a random initialization the eye is completely distorted. However, in a very few iterations, the eye diagram starts being open. In the next iterations, anyway, the convergence starts being slower and slower.

Figure 5.4 shows the histograms of the same iterations taken into account for the eye diagrams. Anyway, after the tenth iteration the effect of the predistortion scheme can be seen. The gaussian corresponding to the same metasymbol starts squeezing as the equalization task does.

This simulation results show that the predistortion scheme using the indirect learning architecture is similar to the equalization at the receiver and for this reason the results seem promising to be applied in an experiment.

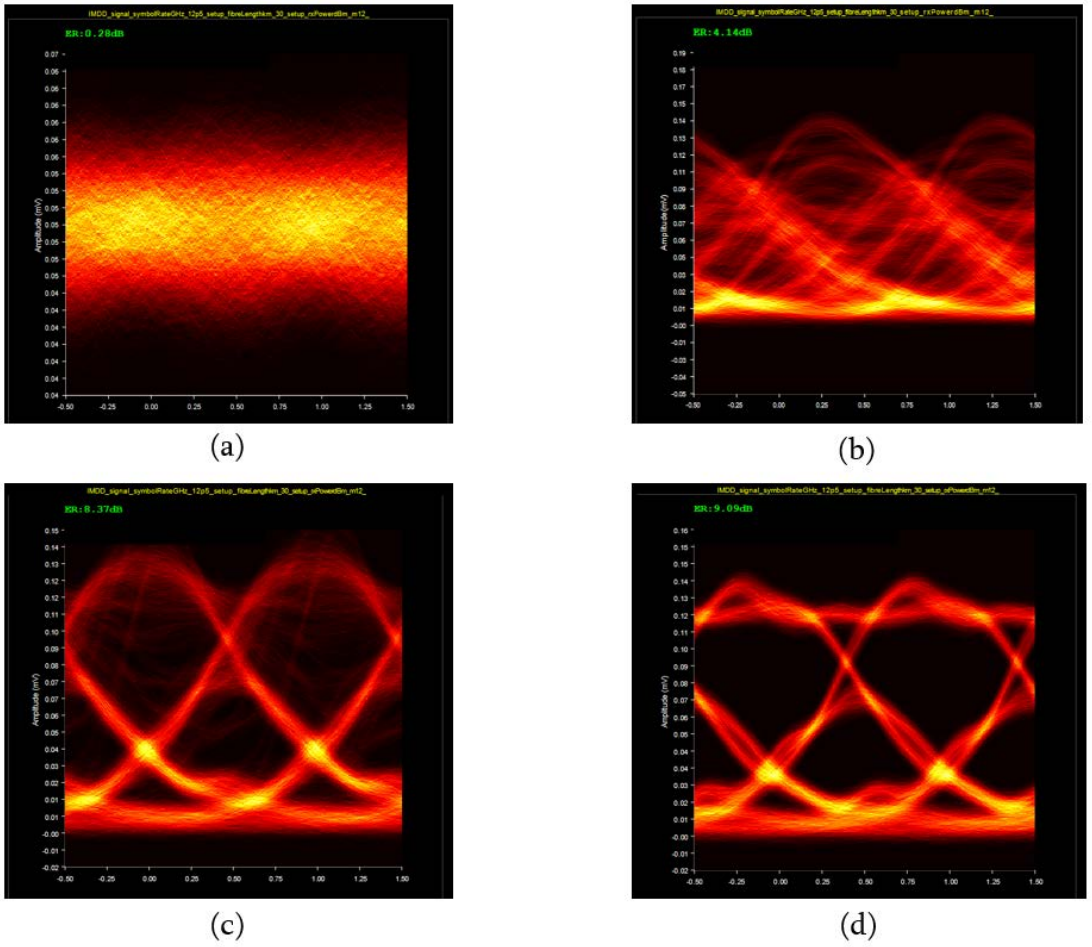


Figure 5.3: Eye diagram at $(\mathbf{a})$ iteration 1, $(\mathbf{b})$ iteration 5, $(\mathbf{c})$ iteration 10 and $(\mathbf{a})$ iteration 15. The transmission length is 30 km, bitrate at 12.5 Gb/s and -12 dBm received power. The neural network has 2 hidden layers with 32 neurons each and sigmoid activation function.

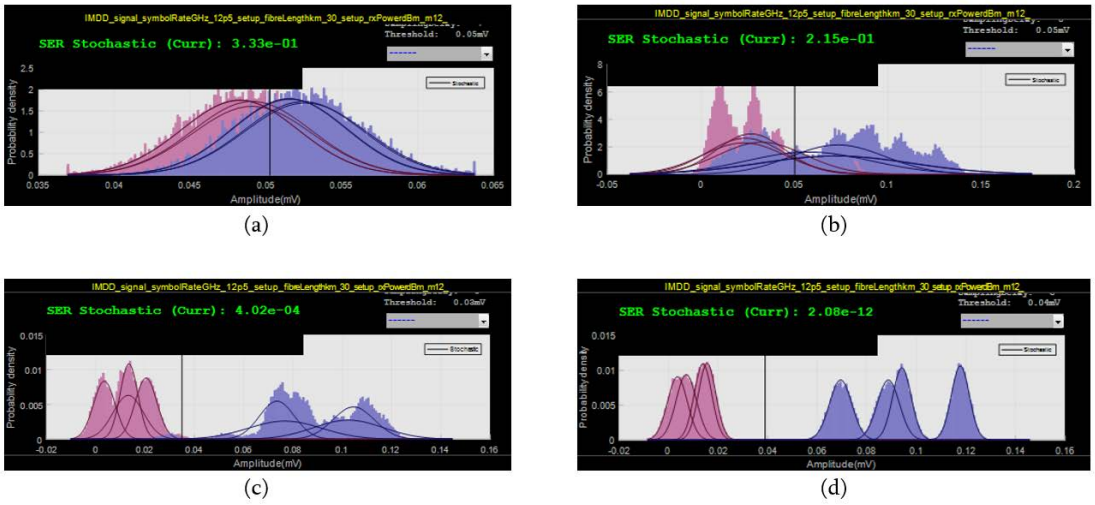


Figure 5.4: Histograms at $(\mathbf{a})$ iteration 1, $(\mathbf{b})$ iteration 5, $(\mathbf{c})$ iteration 10 and $(\mathbf{a})$ iteration 15. The transmission length is 30 km, bitrate at 12.5 Gb/s and -12 dBm received power. The neural network has 2 hidden layers with 32 neurons each and sigmoid activation function.

5.2 PSO for Predistortion

As underlined above, the generation of a predistorted signal is hard because the target signal is unknown. For this reason, a unsupervised learning approach must be applied. In this section, an approach which exploits NNs combined with PSO algorithm is illustrated.

5.2.1 General Overview

As mentioned in the previous chapter, the PSO algorithm can be applied as a tool to optimize the weights of a NN. PSO does not require to have a target in order to adjust the weights of the NNs, but only an optimization function is necessary. In this case, in which the aim is to optimize the performance of the transmission, the BER at the receiver can be a viable measure that can be used as the fitness function of the algorithm.

Both counting and stochastic BER can be used as the fitness function, despite the fact, the former requires enough long sequence to transmit in order to have a statistical guarantee on the measurement which leads to a longer computation. On the other hand, the use of stochastic BER as fitness function corresponds to optimize the opening of the eye diagram and requires a shorter transmission. A combination of both measurements can be exploited to improve the overall speed of computation.

The NNs in this application are exploited to produce power samples of the signal as out-

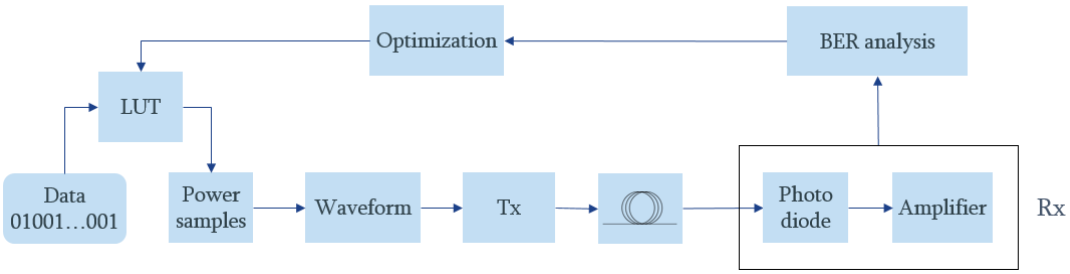


Figure 5.5: Schematic of MATLAB simulation setup for predistorted signal generation using neural networks combined with particle swarm optimization.

put which can be used by an AWG to generate the predistorted signal in an experimental scenario. However, the results that are shown below regards obtained by simulations only, but that can be considered reliable enough to pursue an experiment.

5.2.2 Implementation Details

The experiment is simulated in MATLAB. The simulation captured all the relevant aspects of the transmission: transmitter with the effect of the chirp, optical fibre cable with the non-linearity due to chromatic dispersion, receiver and thermal noise (received power), modulation format and bitrate of the transmission, amount of ISI to take into account, transmitter and receiver bandwidth. Moreover, at the receiver the analysis of the received signal is computed, in order to calculate counting and stochastic BER, used for the update of the particles.

A schematic of the simulation setup is depicted in Figure 5.5.

The bit sequence to transmit is PRBS15. In order to generate the waveform in MATLAB, a look-up table (LUT) strategy is exploited for speeding up the computation. The input of the neural network corresponds to a certain number of taps (3 or 5), according to the ISI to take into account and the output corresponds to a certain number of power samples of the signal to generate. The LUT is built by propogating from the input to the output all the possible combinations of bits. The size of the LUT is given by $2^{\text{ISI}} \times$ `number of power samples`. A graphic representation of the generation of the predistorted signal using a LUT is shown in Figure 5.6.

By means of the LUT, it is not necessary to forward through the NN each sequence of taps which constitutes the whole binary sequence to transmit.

The LUT entries are always numbers in the range $[0.2 - 1.8]$ mW. When the activation function is linear, the output are normalized in this interval. In other cases, the activation

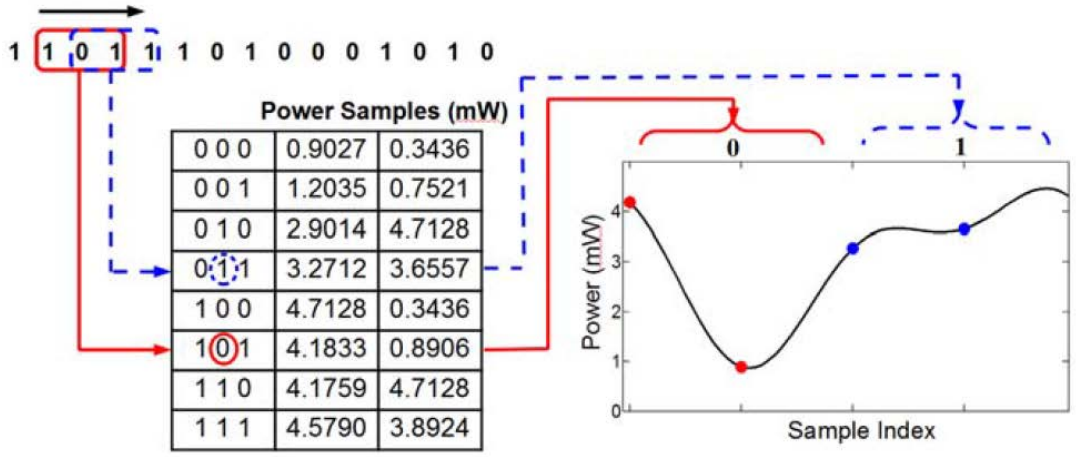


Figure 5.6: Example of predistorted signal generation using a LUT with 8 entries (ISI = 3) and 2 power samples [38].

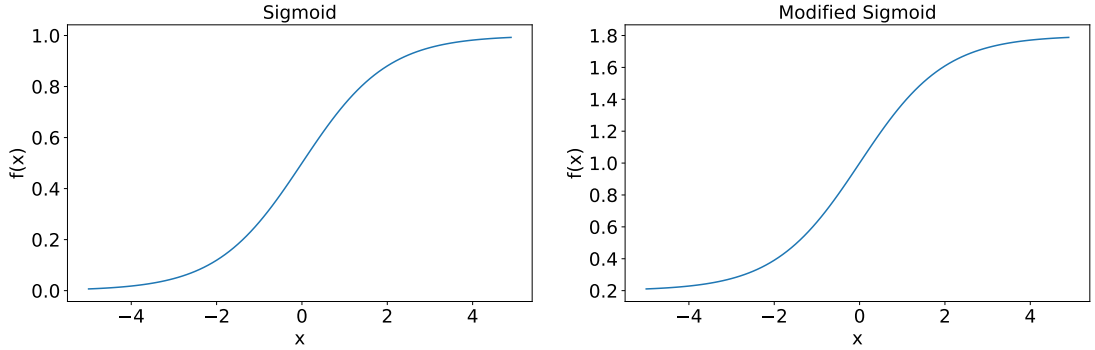


Figure 5.7: Comparison between sigmoid activation function and the modified sigmoid activation function.

function of the output layer is a modified sigmoid function to adapt it to the inteded applications, given by the equation

$$f(x) = \left(\frac{1}{1+e^{-x}} \cdot (d_{max} - d_{min})\right) + d_{min} \tag{5.2}$$

where d_{max} is the upper bound (1.8 mW) and d_{min} is the lower bound (0.2 mW). The comparison between the sigmoid function and modified sigmoid function is shown in Figure 5.7.

The whole simulation has been carried out by using MATLAB for the signal generation, transmission and detection simulation, while the generation of the particles (neural networks), their optimization exploting the PSO algorithm with BER as fitness function and the LUT generation is performed in Python.

The LUT approach is also used to compare the optimization of the predistorted signal generation using the PSO algorithm with MATLAB `fminsearch` optimization algorithm. In this case, the algorithm which is a pattern search optimization, is used to optimize the power samples entries of the LUT and the BER is used as optimization function.

The settings of the parameters for the particle swarm optimization algorithm depend on the transmission length. The longer the transmission length, the more iterations and the more particles are needed for the algorithm to converge because the effects of the distortion are more severe.

5.2.3 Results

The simulation has been run for different transmission lengths, by varying the number of taps which corresponds to the amount of ISI taken into account (3 or 5), the number of power samples used to generate the signal (2 or 4), the bitrate. Regarding PSO algorithm and NNs hyperparameters, different simulations have been run by changing the number of particles, the number of iterations and the inertia weight for PSO, the number of neurons in the hidden layer and the activation functions on the hidden and output layers in the NNs. The modulation format used is NRZ.

Firstly, the results of the PSO algorithm are shown, by comparing the signal generation, eye diagram and the statistical distribution of the metasymbol with and without a predistortion approach. Moreover, a comparison between MATLAB `fminsearch` optimization function and PSO is provided.

Figure 5.8 shows the waveform and the chirp of the signal before and after the optimization using PSO. The simulation setup includes 20 km transmission length, 25 Gb/s bitrate and -12 dBm received power. The PSO algorithm has been run for 100 iterations with 30 particles. The neural networks have 5 neurons in the hidden layer with ReLU activation function and linear activation function in the output layer. The amount of ISI taken into account is equal to 3 and there are 4 output power samples for the signal generation with 32 entries in total in the LUT.

The waveform after the optimizations shows many oscillations especially when the level 0 is transmitted. The behaviour of the chirp is influenced by the transmitted waveform.

Figure 5.9 shows the eye diagram before and after the optimization for the same simulation parameters. As you can see, the eye diagram after the optimization starts being open with an improvement of 3 orders in terms of BER.

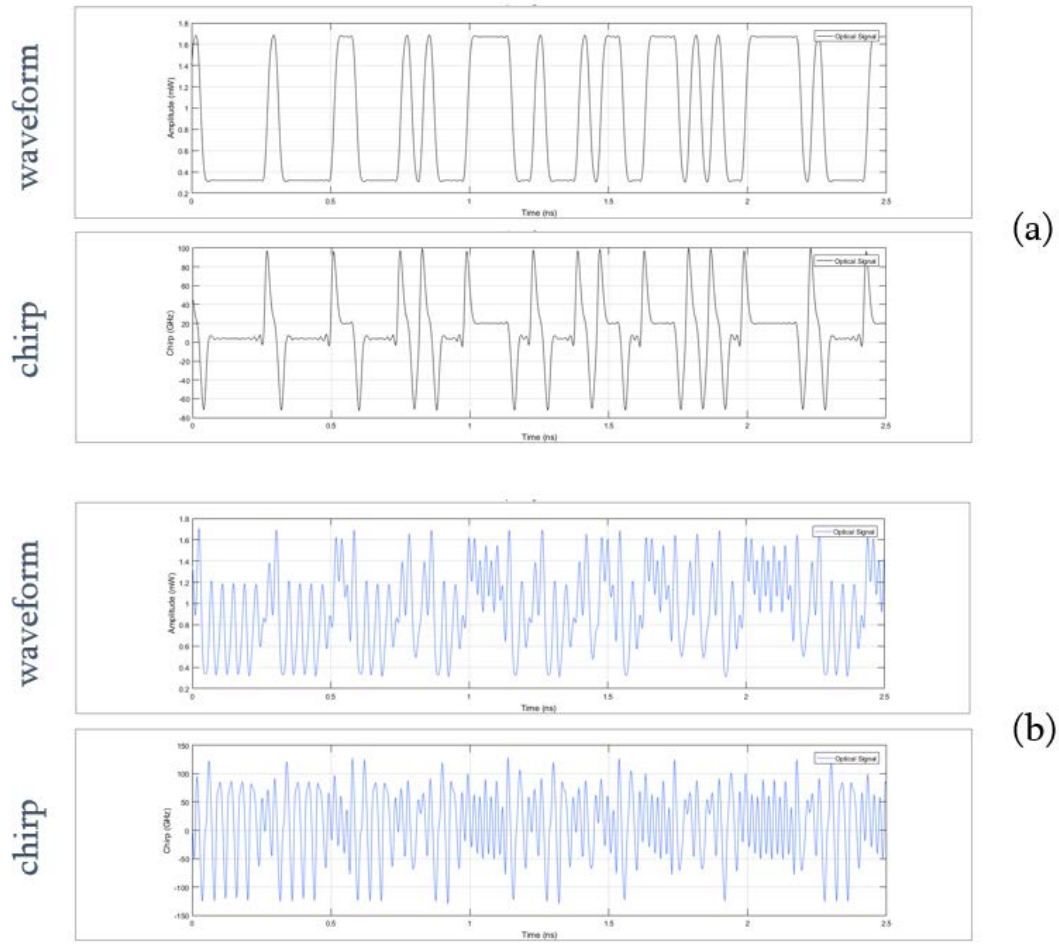
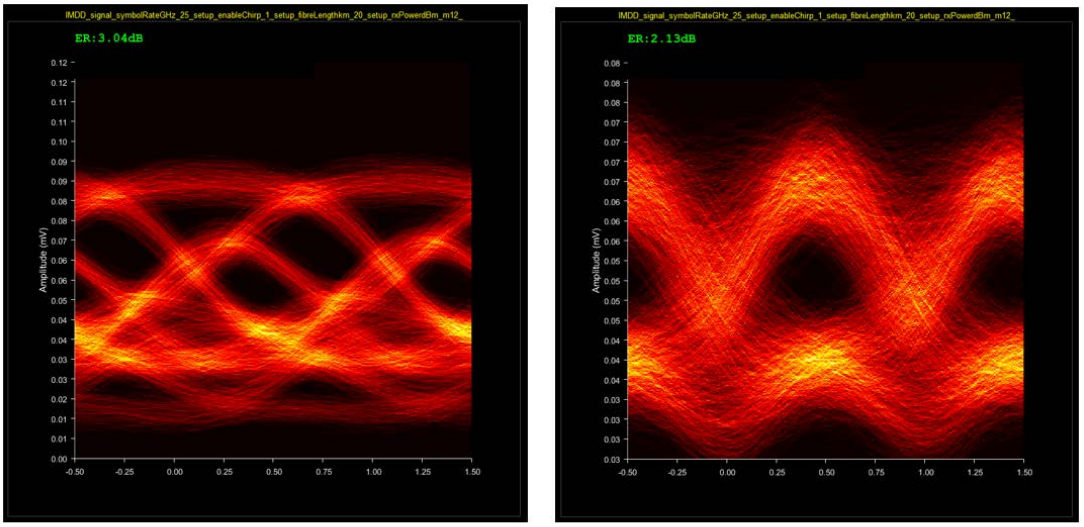


Figure 5.8: Waveform and chirp $(\mathbf{a})$ before and $(\mathbf{b})$ after optimization. The transmission length is 20 km, bitrate at 25 Gb/s and -12 dBm received power. The PSO algorithm has been run for 100 iterations with 30 particles. The neural networks have 5 neurons in the hidden layer with ReLU activation function and linear activation function in the output layer.



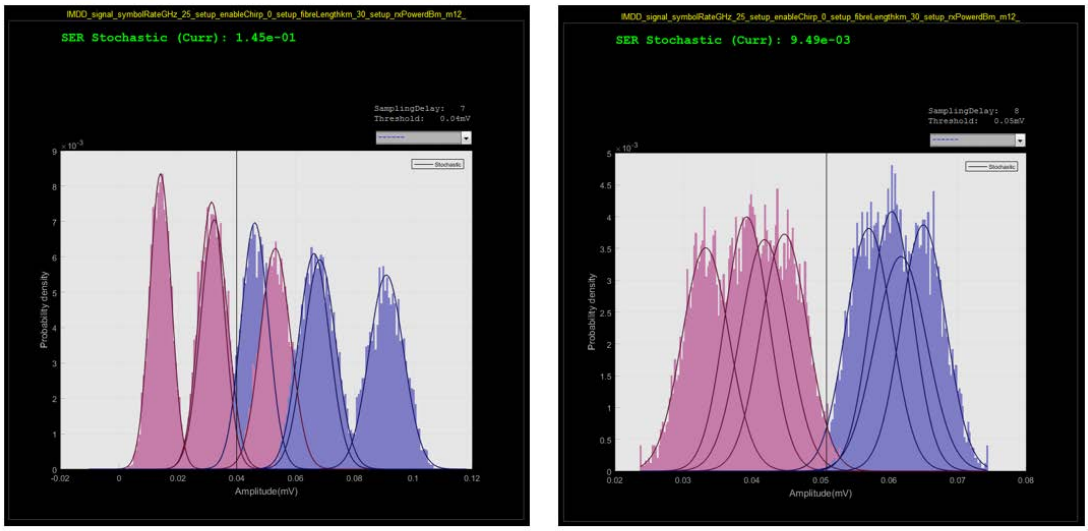
(a) (b)

Figure 5.9: Eye diagram $(\mathbf{a})$ before and $(\mathbf{b})$ after optimization. The transmission length is 20 km, bitrate at 25 Gb/s and -12 dBm received power. The PSO algorithm has been run for 100 iterations with 30 particles. The neural networks have 5 neurons in the hidden layer with ReLU activation function and linear activation function in the output layer.

The effects of the PSO algorithm in terms of perfomance of the transmission can be seen also by looking at the statistical distribution of the metasymbols before and after the optimization. Figure 5.10 shows stochastic BER with and without optimization. The simulation setup includes 30 km transmission length, 25 GHz bitare and -12 dBm received power. The PSO algorithm has been run for 50 iterations with 30 particles. The neural networks have 4 neurons in the hidden layer with sigmoid activation function. The amount of ISI taken into account is equal to 3 and there are 2 output power samples for the signal generation with 16 entries in total in the LUT. In this case, the activation function in the output layer is not the linear function, but the modified sigmoid function in order to limit the output in the range $[0.2 - 1.8]$ mW.

As you can, the effect of the PSO is to reduce the ISI of the transmission which affects the overall performance. The error region is heavily reduced by limiting the overlap between the gaussians which correspond level 0 to ones which correspond to level 1.

As mentioned below, the optimization of the LUT has been perfomed also by running MATLAB `fminsearch` optimization algorithm. While for NNs, the LUT is initialized based on the ranodm initialization of the network, for `fminsearch` the LUT is initialized



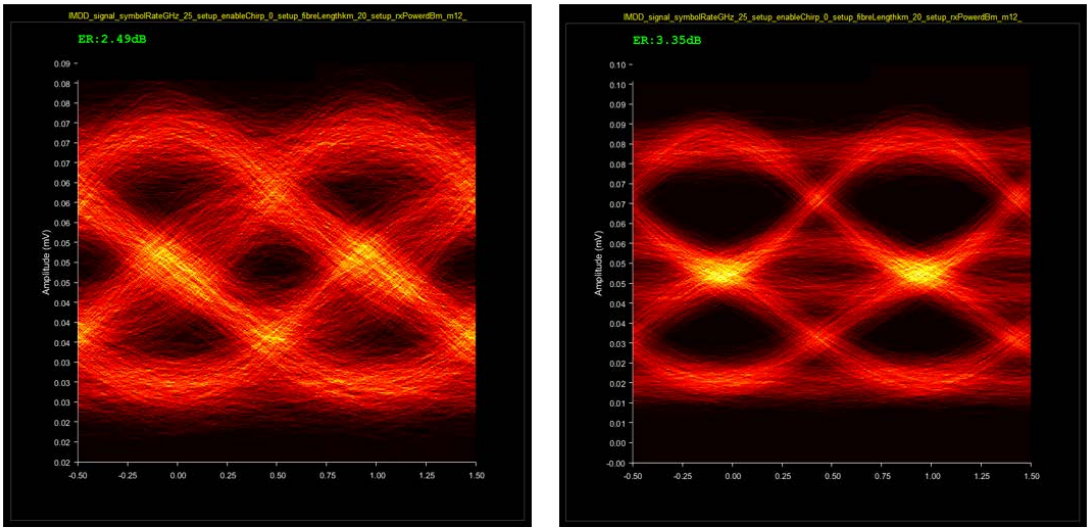
(a) (b)

Figure 5.10: Stochastic BER $(\mathbf{a})$ before and $(\mathbf{b})$ after optimization. The transmission length is 30 km, bitrate at 25 Gb/s and -12 dBm received power. The PSO algorithm has been run for 50 iterations with 30 particles. The neural networks have 4 neurons in the hidden layer with sigmoid activation function and and modified sigmoid activation function in the output layer.

by sampling an ideal waveform, smoothed by a Bessel filter. Figure 5.11 shows the comparison on the eye diagram after the optimization by running both PSO and `fminsearch` on transmission of a binary sequence with 25 Gb/s bitrate, transmission length of 20 km and received power of -12 dBm. The LUT considers 5 taps and 2 power samples for the signal generation. `fminsearch` is run for 800 iterations, while PSO is run for 100 iterations and 30 particles, with 3 neurons in the hidden layer and ReLU activation function. The activation function in output layer is the linear function.

As you can see, despite the fact the eye is still distorted with both algorithms after the optimization, PSO performs better. Moreover, the eye diagram in the figure is achieved by `fminsearch` function after about 100 iterations, while up to the iteration 800 is around the same local minimum. This explains that `fminsearch` function is not suitable for this optimization problem because the performance strictly depends on the initialization of the LUT and the algorithm is trapped in the closest local minimum it finds. On the other hand, a numerous number of particles are able to explore a bigger hyperspace without being trapped in a local minimum.

The results depend on the initialization of the particles which is random. However, this affects only the time of the algorithm to converge, but the overall performance after the opti-



(a) (b)

Figure 5.11: Eye diagram after optimization with $(\mathbf{a})$ PSO algorithm and $(\mathbf{b})$ MATLAB `fminsearch` function. The transmission length is 30 km, bitrate at 25 Gb/s and -12 dBm received power. MATLAB `fminsearch` function has been run for 800 iterations, while the PSO algorithm has been run for 100 iterations with 30 particles. The neural networks have 3 neurons in the hidden layer with ReLU activation function and linear activation function in the output layer.

mization is similar. Anyway, the results of the PSO algorithm seem promising to be applied in an experiment.

Variants of the PSO algorithm like democratic PSO or adaptive methods could be used in order to improve the convergence time. Furthermore, a different approach for the generation of the signal which consists in replacing the metasymbols as the input of the NN by using delayed taps of the ideal signal waveform, smoothed by a filter, based on the binary sequence to transmit can be a viable solution, as shown by Stefan Warm in [39]. In this case, the LUT approach cannot be used because there is no fixed number of entries anymore, but real values depending on the ideal waveform.

6

Conclusion and Future Work

In this work, two predistortion scheme are presented. In particular, the indirect learning architecture using neural networks is applied for the first time to optical communication using a directly modulated laser.

Motivation on the effectiveness of neural networks applied to generate a predistorted signal was given in Chapter 3. The results of the regression task showed the ability of NNs as universal approximator.

In Chapter 4, we focused on the use of PSO as an optimization algorithm for NNs. The experiment showed that, even though the tuning of the hyperparameter and the initialization of the particles affect the convergence of the algorithm, PSO allows NNs to reach similar performance respect to the traditional training algorithm like backpropagation.

In Chapter 5, two predistortion schemes have been introduced and the results on the simulation have been reported. While the PSO requires much time to the convergence, the ILA showed promising results in a very few iterations. Moreover, the ILA with NNs applied to the optical communication to precompensate the chromatic dispersion with DML represents the novelty of this work.

After the results demonstrated in simulation, future work includes the experimental results using the predistortion schemes.

Concerning the PSO algorithm, the experiment can be carried out in practice for a computational point of view only with a modification of the optimization phase. Because the whole transmission and the direct detection of the signal at receiver requires a few minutes,

the high number of iterations and the high number of particles leads the experiment to be impractical. For this reason, a modified version in which each transmitted signal is composed of packets where each packet is generated by a different neural network and with a sufficient number of bit to allow the results to be reliable should be implemented in order to carry out one iteration of the algorithm with only one transmission. Moreover, other optimization algorithms which include genetic algorithms can be implemented in order to compare the performance.

Further implementations and future works include online training of the network to adapt the networks to react to changes of the setup and the comparison of the performance of the transmission with the combination of NN-based predistorter at the transmitter and a NN-based equalizer at the receiver.

References

[1] P. J. Winzer and R. . Essiambre, "Advanced optical modulation formats," *Proceedings of the IEEE*, vol. 94, no. 5, pp. 952–985, May 2006.

[2] N. Benvenuto and M. Zorzi, *Principles of communications networks and systems*. Chichester, West Sussex, United Kingdom: John Wiley and Sons Ltd, 2010.

[3] S. Meroli, "Optical fiber vs electrical cable communications." [Online]. Available: http://meroli.web.cern.ch/lecture_fibre_vs_copper.html

[4] G. P. Agrawal, *Optical Communication: Its History and Recent Progress*. Cham: Springer International Publishing, 2016, pp. 177–199. [Online]. Available: https://doi.org/10.1007/978-3-319-31903-2_8

[5] D. E. Borth, W. E. Stark, and J. S. Lehnert, "Telecommunication," 2019. [Online]. Available: https://www.britannica.com/technology/telecommunication

[6] K. Grobe and M. Eiselt, *Wavelength Division Multiplexing: A Practical Engineering Guide*, 1st ed. Wiley Publishing, 2013.

[7] M. H. Weik, *space-division multiplexing*. Boston, MA: Springer US, 2001, pp. 1623–1623. [Online]. Available: https://doi.org/10.1007/1-4020-0613-6_17780

[8] F. Effenberger, D. Cleary, O. Haran, G. Kramer, R. D. Li, M. Oron, and T. Pfeiffer, "An introduction to pon technologies [topics in optical communications]," *IEEE Communications Magazine*, vol. 45, no. 3, pp. S17–S25, March 2007.

[9] "Optical fibres." [Online]. Available: http://macao.communications.museum/eng/exhibition/secondfloor/MoreInfo/2_8_3_OpticalFibres.html

[10] R. Paschotta, "article on "optical fiber communications" in the rp photonics encyclopedia." [Online]. Available: https://www.rp-photonics.com/optical_fiber_communications.htmll

[11] W. Mcculloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 127–147, 1943.

[12] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *1960 IRE WESCON Convention Record, Part 4*. New York: IRE, 1960, pp. 96–104.

[13] F. Rosenblatt, *Principles of neurodynamics: perceptions and the theory of brain mechanisms*. Washington, DC: Spartan, 1962. [Online]. Available: https://cds.cern.ch/record/239697

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362. [Online]. Available: http://dl.acm.org/citation.cfm?id=104279.104293

[15] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. New York, NY, USA: Cambridge University Press, 2014.

[16] Y. Ge, R. Zhang, W. Lingyun, X. Wang, X. Tang, and P. Luo, "Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images," 01 2019.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[18] M. Lučić, M. Ritter, M. Tschannen, X. Zhai, O. F. Bachem, and S. Gelly, "High-fidelity image generation with fewer labels," in *International Conference on Machine Learning*, 2019. [Online]. Available: https://arxiv.org/abs/1903.02271

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. [Online]. Available: http://www.nature.com/articles/323533a0

[20] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[22] N. Srivastava, "Improving Neural Networks with Dropout," Master's thesis, University of Toronto, Toronto, Canada, January 2013.

[23] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jul. 1989. [Online]. Available: http://dx.doi.org/10.1016/0893-6080(89)90020-8

[24] T. Eriksson, H. Bulow, and A. Leven, "Applying neural networks in optical communication systems: Possible pitfalls," *IEEE Photonics Technology Letters*, vol. PP, pp. 1–1, 09 2017.

[25] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989. [Online]. Available: http://dx.doi.org/10.1162/neco.1989.1.4.541

[26] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*. London, UK, UK: Springer-Verlag, 1999, pp. 319–. [Online]. Available: http://dl.acm.org/citation.cfm?id=646469.691875

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.

[29] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

[30] K. Tehrani, Y. Zhang, P. Shen, and P. Kner, "Adaptive optics stochastic optical reconstruction microscopy (ao-storm) by particle swarm optimization," *Biomedical Optics Express*, vol. 8, p. 5087, 11 2017.

[31] W. T. Reeves, "Particle systems—a technique for modeling a class of fuzzy objects," *ACM Trans. Graph.*, vol. 2, no. 2, pp. 91–108, Apr. 1983. [Online]. Available: http://doi.acm.org/10.1145/357318.357320

[32] A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, 01 2017.

[33] Changsoo Eun and E. J. Powers, "A new volterra predistorter based on the indirect learning architecture," *IEEE Transactions on Signal Processing*, vol. 45, no. 1, pp. 223–227, Jan 1997.

[34] Lei Ding, G. T. Zhou, D. R. Morgan, Zhengxiang Ma, J. S. Kenney, Jaehyeong Kim, and C. R. Giardina, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Transactions on Communications*, vol. 52, no. 1, pp. 159–165, Jan 2004.

[35] M. Schetzen, *The Volterra and Wiener theories of nonlinear systems / Martin Schetzen*. Wiley New York, 1980.

[36] Changsoo Eun and E. J. Powers, "Utilization of neural network signal processing in the design of a predistorter for a nonlinear telecommunication channel," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 6, June 1994, pp. 3582a–3586 vol.6.

[37] N. Benvenuto and F. Piazza, "On the complex backpropagation algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 967–969, April 1992.

[38] A. S. Karar, M. Yanez, J. Cartledge, J. Harley, and K. Roberts, "Electronic dispersion pre-compensation using a directly modulated laser at 10.7-gb/s," 01 2011.

[39] S. Warm, C.-A. Bunge, T. Wuth, and K. Petermann, "Electronic dispersion precompensation with a 10-gb/s directly modulated laser," *Photonics Technology Letters, IEEE*, vol. 21, pp. 1090 – 1092, 09 2009.

Acknowledgments

First of all, I would like to thank my mother Marinella and my sisters Ilaria, Laura and Claudia, my brothers-in-law and my nephews for their patience and support during these years at the university.

I would like to thank my supervisors Dr Cleitus Antony, Professor Paul Townsend and Professor Fabio Vandin for their continued support and the motivation they gave me and Professor Leonardo Giudicotti who found me this project in Cork.

I also would like to thank Stephen Murphy from the Photonics Systems Group at Tyndall National Institute for his help and his advice. It was a real pleasure to work with you.

Finally, I can't forget all my friends and my colleagues at Università degli Studi di Padova and at Tyndall National Institute for their constant presence in supporting me when I needed help and with who I spent memorable moments.

Financial support for this work is gratefully acknowledged from Tyndall National Institute and Science Foundation Ireland (Project SFI PI Award R14966).