UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

MASTER THESIS IN ICT FOR INTERNET AND MULTIMEDIA

Pulse shaping optimization via machine learning for optical fiber transmission

MASTER CANDIDATE

Tullia Fontana

Student ID 2058909

SUPERVISOR

Nicola Laurenti

University of Padova

CO-SUPERVISOR

Francesco Da Ros

Technical University of Denmark

ACADEMIC YEAR 2022-2023

SEPTEMBER 5, 2023

To my family.

Abstract

In this work, we propose a non-conventional framework based on machine learning (ML) able to provide equalization capabilities in a communication channel with impairments. Traditional Pulse Shaper/Matched Filter blocks will be replaced by trainable FIRs aimed at the optimization of the pulse shaping in order to compensate for channel impairments in absence of a traditional equalization block. The specific application case that motivated this work is the short-reach optical link; on this scenario we will base the literature review and the problem statement. Then, in the simulations, the impaired channel we tested is a band-limited AWGN channel whose perfect knowledge allowed us to provide a benchmark for the obtained results, achieving the goal of validating the proposed ML-based method. In the view of a more widespread adoption of ML in communications, this work can be considered as a methodological evaluation of a non-conventional ML-based equalization method, aimed at exploring the opportunities that ML can enable on the Pulse Shaper/Matched Filter blocks and, in principle, can be extended to more complex channels where the optimal solutions have not been found yet - like the optical channel - providing effective improvements in terms of performance and complexity.

Sommario

Contents

List of Figures

List of Tables

List of Acronyms

ML Machine Learning

PS Pulse Shaper

MF Matched Filter

FIR Finite Impulse Response

RRC Square Root-Raised Cosine

AWGN Additive White Gaussian Noise

NLSE Nonlinear Schrödinger equation

CD Chromatic Dispersion

GVD Group velocity Dispersion

DD Direct Detection

RC Root-Raised Cosine

PD Photodetector

ISI Intersymbol Intereference

FFE Feed Forward Equalizer

ZF Zero Forcing

LMS Least Mean Square algorithm

RLS Recursive Least Square algorithm

DFE Decision Feedback equalizer

VNLE Volterra Nonlinear Equalizer

FFNN Feed Forward Neural Network

NN Neural Network

BRNN Bidirectional Recurrent Neural Network

DSP Digital Signal Processing

SER Symbol-Error-Rate

SE Spectral Efficiency

GPU Graphical Processing Unit

SGD Stochastic Gradient Descent

MSE Mean Squared Error

SNR Signal-to-Noise Ratio

I In-Phase

Q Quadrature

PSD Power Spectral Density

MSGD Mini-batch Gradient Descent

SPSA Simultaneous Perturbation Stochastic Approximation

LPF Low-Pass Filter

MPSK M-ary Phase Shift Keying

LTI Linear Time-Invariant

WDM Wavelength Division Multiplexing

ICI Interchannel Interference

1 Introduction

1.1 Context and Motivation

Due to the physical nature of the transmission media and to non-ideal hardware devices in communication systems, the propagation along the communication channel causes the distortion of the transmitted signal with severe performance penalties at the receiver.
In the last decades, communication engineers have developed various methods to perform the equalization of channels, achieving low error probability, high spectral efficiency and low latency in various use cases. The most common equalization techniques include digital pre-distortion and post-compensation and rely on a formal definition of the channel model and on robust algorithms that allow to compensate - or at least to mitigate - for the channel impairments. Currently, these methods are well-established and achieve remarkable results or, in some cases, even approach the optimal solution defined by the physical limits of the considered channel [32].
However, despite the vast expert knowledge in this field, there are still scenarios in which the current state-of-art does not provide an optimal compensation. There are several reasons that may obstacle the achievement of the optimal performance. For instance, in some scenarios, the problem of interest does not have an analytical solution and, thus, it is not possible to adopt traditional techniques that rely on the inversion of the channel model. Or, in other cases, the algorithm to find the solution for the problem of interest is too much complex to be implemented for the target scenario, especially for real-time implementations [4].

Since the traditional methods are not applicable in the aforementioned cases, to address the dramatic increase of the demand for high data rates, the research field is seeking novel non-conventional solutions to improve the performances of communication systems.
One promising direction can be the adoption of machine learning (Machine Learning (ML)) [56].
Exploiting the large amount of data produced by network elements, ML would enable the development of automatable software and in principle could achieve better performances than classic algorithms [7].
As a result, even in absence of an analytical solution for a problem of interest, ML would be able to provide an approximated solution and, in addiction, it would allow the development of novel algorithms with a lower level of complexity, going beyond the current state-of-art, providing improvements in terms of performance and complexity.
However, differently from other fields like computer vision or natural language processing [16] where the role of machine learning is well-established, ML in communications is still at its infancy and, to enable its widespread adoption, a novel infrastructure must be developed before network data can be effectively consumed by ML applications [7] .

On this path, in this work, we investigated a specific scenario that is affected by impairments where the use of ML can enable a performance improvement: the short-reach optical link. First, we analyzed the sources of impairment that affect the short-reach optical link causing linear and non linear distortions, in order to understand why ML can provide a performance enhancement. Then, after a literature review in which we analyzed the most widely used equalization methods in this specific scenario, we focused on the recent research direction that proposes ML-based solutions to perform the equalization of the received signal.
Among them, solutions based on the optimization of the Pulse Shaping block caught our attention and, since there was a low number of studies on this specific task, we decided to further investigate on it.
More specifically, in this work, we developed a ML-based framework with trainable FIR filters that we placed instead of conventional Pulse Shaper (PS) and/or the Matched Filter (MF) blocks resulting into three different scenarios. In the first, the trainable Finite Impulse Response (FIR) filter is placed at the transmitter and is paired with a Square Root-Raised Cosine (RRC) at the receiver, that is

one of the most widely used pulse shape; in the second, we had a FIR filter at the receiver and a RRC at the transmitter; finally, in the third, both the PS and MF are trainable and we decided to optimize them jointly, adopting the so-called end-to-end learning that is a novel technique we found in recent literature [32]. The objective is to train the FIRs filters in order to learn the proper pulse shapes to provide equalization functionalities in a channel with impairments. In other words, we want to learn the FIR taps able to compensate for the distortion induced by the channel without implementing a traditional equalizer.
Because of the inexperience, in this work, we didn't have time to test the short-reach optical link that we had investigated in the literature and that gave rise to this specific ML-based implementation. The impaired channel that we actually implemented in this work was the band-limited Additive White Gaussian Noise (AWGN) channel. Even though the channel in question does not belong to the use cases that motivate the use of ML, this work can be considered as a preliminary study to investigate the opportunity enabled by a non-conventional ML-based equalization method in a channel with impairments. In this case, the source of impairment will be the limited bandwidth that introduces intersymbol interference for which an equalizer is usually needed.
Therefore in this work, we will train our FIRs in the three scenarios we mentioned in order to compensate for the penalties of the band-limited AWGN channel.
To validate our results, given that the band-limited AWGN channel allows us to easily develop conventional equalization methods, we implemented traditional post-equalizers at sample-level and also at symbol-level. As a result, we obtained a benchmark for the outcomes of the proposed ML-based framework.
To sum up, this work consists in a methodological evaluation of a non conventional ML-based equalization method that we tested and validated on the band-limited AWGN channel.
In future works, the proposed framework can be extended to more complex channel scenarios in order to exploit the potential that ML can enable in principle and that we can not appreciate in the considered case where the optimal solution can be achieved with traditional equalization methods.

1.2 Organization of the thesis

The work will be organized as follows.
In Chapter 2, we described a specific application case in which ML can provide an improvement in terms of performances that is the short-reach optical link that motivated this work.
In Chapter 3, we presented both traditional and novel ML-based equalization methods for the short-reach optical link. In addiction, we summarized recent ML-based solutions for the optimization of the PS and MF blocks, extending our view to optical communications in general because of the limited amount of works that deal with this specific topic.
In Chapter 4, we discussed more in details the opportunity to apply ML to communication networks, understanding how it is different to traditional methods and how it can provide significant improvements.
In Chapter 5, we presented the problem that we analyzed in this work from a general point of view.
In Chapter 6, after a brief introduction about ML, we presented the ML tools that we used in this work.
In Chapter 7, we presented the setup of the band-limited AWGN channel we analyzed, proposing conventional equalization techniques as a benchmark.
In Chapter 8, we described the non-conventional ML-based framework we implemented to perform the equalization of channel impairments and we compared them with the developed conventional post-equalizers solutions.
In Chapter 9, we summarized this work and underlined the novelties of our contribution. Proposals for future works conclude the thesis.

2

Application: the short-reach optical link

In the last decades, in the context of optical communications, short-reach optical links systems have received the attention from both industry and academia. The term short-reach is used to refer to communication systems and networks with a transmission distance shorter than 100 km [62]. Use cases include intra-datacenter and inter-datacenter networks, optical access networks, indoor optical wireless communications, etc. [62]. The demand for high data rates in these applications is increasing and there are several performance limiting factors that need to be addressed. These optical connections have indeed stringent requirements in terms of complexity, cost, power consumption [62]. In addiction, the optical fiber properties interact with hardware limitations, causing performance degradation.

To address these issues, ML has started to be involved in the equalization process, providing effective solutions to mitigate the impairments that affect the short-reach optical channel.

In this work, our problem statement and scenario were inspired on the problem arisen by this specific scenario. In the next Section, we will describe the channel model and then the transceiver model, introducing non-idealities that we have to consider when we deal with a short-reach optical communication system.

2.1 Short-reach optical channel model

The propagation of the light in optical fiber is governed by the Nonlinear Schrödinger equation (NLSE) [1] that describes physical phenomenons that may affect the fiber, i.e. attenuation and Chromatic Dispersion (CD), both linear effects, the nonlinear Kerr-effect, etc. It can be expressed as follows:

$$\begin{aligned} \frac{\partial A}{\partial z} &= -\frac{\alpha}{2}A && \text{Linear Attenuation} \\ &- j\frac{\beta_2}{2}\frac{\partial^2 A}{\partial t^2} && \text{Second Order Dispersion} \\ &+ \frac{\beta_3}{6}\frac{\partial^3 A}{\partial t^3} && \text{Third Order Dispersion} \\ &- j\gamma|A|^2 A && \text{Kerr Effect} \end{aligned} \tag{2.1}$$

In short-reach links, with the proper assumptions, it is possible to neglect several impairments like nonlinearities and Third-Order Dispersion.
In addiction, in this analysis, we decided to focus our attention and to consider only the Second Order Dispersion, i.e. the CD, similarly as we found in [33].
As a result, we assume to model the optical fiber with the following partial differential equation:

$$\frac{\partial A}{\partial z} = -j\frac{\beta_2}{2}\frac{\partial^2 A}{\partial t^2} \tag{2.2}$$

where A is the complex amplitude of the optical field envelope, t is the time, z represents the position along the fiber and β_2 is the dispersion coefficient.
It is possible to solve equation 2.2 analytically in the frequency domain. Thus, we take the Fourier transform and we get the dispersion frequency domain transfer function:

$$D(z,\omega) = \exp\left(j\frac{\beta_2}{2}\omega^2 z\right) \tag{2.3}$$

where ω is the angular frequency.
Because of CD, different spectral components of the transmitted signal propagate at different speeds in the optical channel. As a result, each transmitted pulse broadens and interferes with the adjacent pulses. The velocity at which the envelope of the transmitted pulse propagates is called the group velocity that experiences the so-called Group velocity Dispersion (GVD).

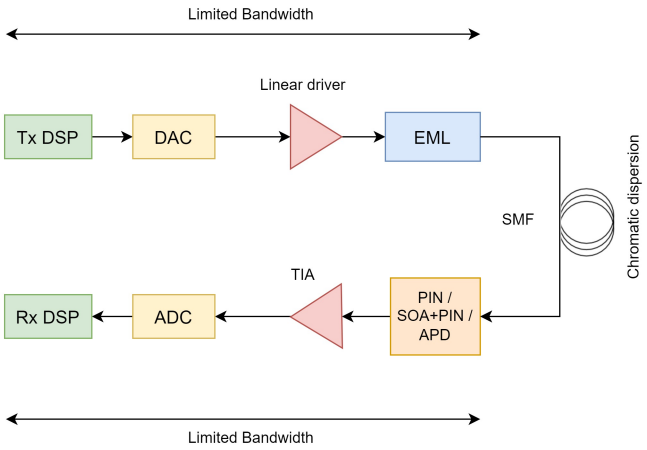


Figure 2.1: Short reach link and possible sources of impairments.

2.2 Limitations in short-reach optical links

Apart from CD - that is a source of impairment that is intrinsic to the optical fiber - there are other effects that degrade a signal that propagates through short-reach optical links. They are caused by both optical and electronic devices. In this section, we will briefly present them.
In short optical links, there are strict constraints in terms of cost. As a result, the devices that are employed in the communication links have various non-idealities that degrade the quality of the received signal. In Figure 2.1, we can see a block diagrams of short-reach optical links in which we represent linear and nonlinear effects that occur along the propagation of a transmitted signal. The two effects that we will consider in this analysis are:

- the linear bandwidth limitations due to the low-cost hardware devices at the receiver;
- the nonlinear interaction between Chromatic Dispersion and the Direct Detection performed by the Photodetector.

In the next sections we will explain more in details such impairments.

2.2.1 Bandwidth limitations

The low-cost implementation in short-reach optical links results into the employment of devices with several non-idealities. In this section, we focus on the bandwidth limitation, that is the channel impairment that we have analyzed in this work.

Given x_k the transmitted signal and y_k the received signal - processed by a band-limited device - we can express the relationship between them with the following formula:

$$y_k = I_k + \sum_{n=0,\, n\neq k}^{\infty} I_n x_{k-n} + v_k \tag{2.4}$$

where k is the instant at which the signal is sampled.
$\{I_n\}_{n\in\mathcal{Z}}$ is the sequence of transmitted symbols and I_k represents the desired information symbol at the k-th sampling instant.
The second term in the equation represents the Intersymbol Intereference (ISI) that consists in an amplitude and phase distortion of the transmitted signal that will cause a performance degradation at the receiver.
Finally, v_k is the sampled version of the additive white Gaussian noise.

2.2.2 Nonlinear interaction between CD and DD at the Receiver

As we discussed, because of CD, every portion of the transmitted pulse spectrum experiences a different delay and, thus, the signal broadens causing intersymbol interference.
In addiction, short-reach optical links are commonly implemented with the Direct Detection (DD) method that performs the detection relying on the signal amplitude, loosing the phase information at the receiver. It is a technologically simple and cost-effective approach that is widely used for instance in intra-datacenter communication with a coverage up to tens of kilometers [57].
The combination of CD - that causes ISI - with direct detection - that does not provide any knowledge about the signal phase - produces a power-fading effect [63] that makes the overall channel through which the signal propagates nonlinear. For this reason, the compensation of CD is a significant challenge that has to be carefully considered in the design of short-reach optical links.

The power-fading effect causes notches, i.e. spectral zeros in the spectrum of the signal. In Figure 2.2, we can see the magnitude response of a 28 Gbaud system with different fiber lengths (15 km, 50 km and 100 km respectively) [37]. As the fiber length increases, there will be more and more spectral zeros, from having none to one and then to three. Observing the Figure, we can notice that in the high frequency the probability of frequency notches is greater. This means that at high-speed transmission the power-fading effect is stronger and, thus, it

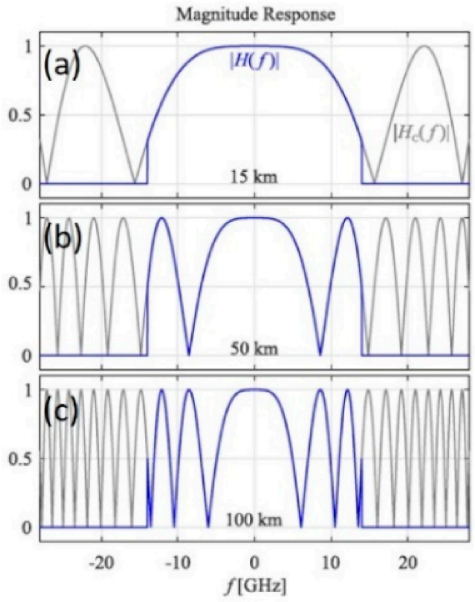


Figure 2.2: Magnitude responses of a 28 Gbaud system at (a) 15 km; (b) 50 km; (c) 100 km in simulation [37].

set a limit to the optical fiber transmission distance.

2.3 A SIMPLE SHORT-REACH OPTICAL LINK MODEL

In this section, we will present a simple setup of a short-reach optical fiber scenario with Direct Detection (DD). In Figure 2.3, we can see a simple block diagram that represents a short-reach optical link. Compared to Figure 2.1, we will consider only Chromatic Dispersion, the presence of a Photodetector and a band-limited hardware device placed at the receiver as sources of impairments. In addiction, we have underlined the presence of PS and MF blocks, that are our focus in this work.
Let us describe more in detail the system model.

1) Transmitter: Let us have a sequence of independent and identically distributed (i.i.d.) symbols $(X_\kappa)_{\kappa\in\mathcal{Z}} = (..., X_1, X_2, X_3, ...)$ with finite alphabet $\mathcal{A} = \{a_1, ..., a_Q\}$ of cardinality Q and symbol probability mass function $p(x) = \frac{1}{Q}$ for all $x \in \mathcal{A}$. Then we have a pulse shape filter with $g_{tx}(t)$; we convolve it with the

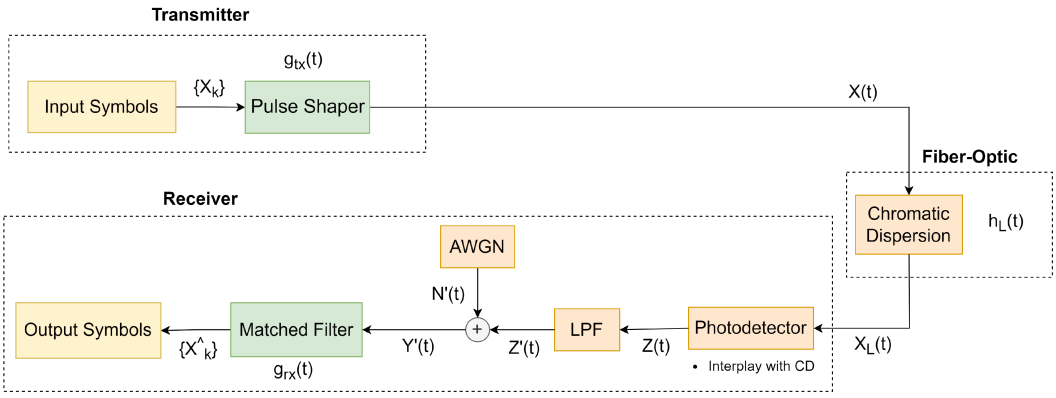


Figure 2.3: Simplified short-reach communication system.

symbol sequence and we get the baseband waveform:

$$X(t) = \sum_{\kappa} X_{\kappa} \cdot g_{tx}(t - \kappa T_s) \tag{2.5}$$

where we have T_s, the symbol-period, and $B = \frac{1}{T_s}$, the symbol-rate.
A common choice for the PS is the RRC [11].

2) Fiber-Optic Link: The channel exhibits CD with response [34]:

$$h_L(t) \rightarrow H_L(\omega) = \exp(j\frac{\beta_2}{2}\omega^2 L) \tag{2.6}$$

where β_2 is the CD parameter, $\omega = 2\pi f$ is the angular frequency and L the length of the fiber.

3) Receiver At the receiver, there are both optical and electrical components. First, there is a Photodetector (PD) that captures the impinging field and returns the intensity $Z(t) = |X_L(t)|^2$ [34]. Afterwards, there is a hardware device that will set a bandwidth constraint $g_{LPF}(t)$ and introduce noise $N'(t)$ that we model as a Gaussian random process with power spectral density $\frac{N_0}{2}$. As a result, its output will be:

$$Y(t)' = Z(t) * g_{LPF} + N'(t) = Z'(t) + N'(t). \tag{2.7}$$

Finally, we have the Matched Filter [34] that will be a RRC equal to the RRC at the Pulse Shaper.

Therefore, the received signal is $Y(t)$:

$$Y(t) = Y'(t) * g_{rx}(t). \tag{2.8}$$

that will be sampled at a rate $\frac{1}{T_s'}$ (the oversampling factor is $sps = \frac{T_s}{T_s'}$ samples per symbol).
To sum up, the overall channel is affected by linear impairments, such as CD and the limited bandwidth of hardware devices at the receiver, and also nonlinear effects, that are due to the interaction of CD with the effect of the PD.

3

Literature review: the road towards an alternative pulse shaping optimization

3.1 Conventional Equalization methods in the short-reach optical link

In order to mitigate the impairments in short-reach optical links, several digital equalization technologies have been proposed. In the next section, we will introduce two of the most common equalizers that are employed to compensate for these impairments. We will focus on the description of the first, the feed-forward equalizer that works in a linear regime and, thus, we can directly relate it with the framework that we implemented in this work. Whereas, for the latter, i.e. Volterra Nonlinear Equalizer (VNLE), we will provide only some hints for a high-level description. Afterwards, more recent ML-based solutions will be described.

3.1.1 Conventional FFE/DFE

The Feed Forward Equalizer (FFE) is a widely used method to compensate for linear impairments. The most important component of FFE is the FIR filter that we can see in Figure 3.1a. The FIR acts on a given input $x(k)$ and generate

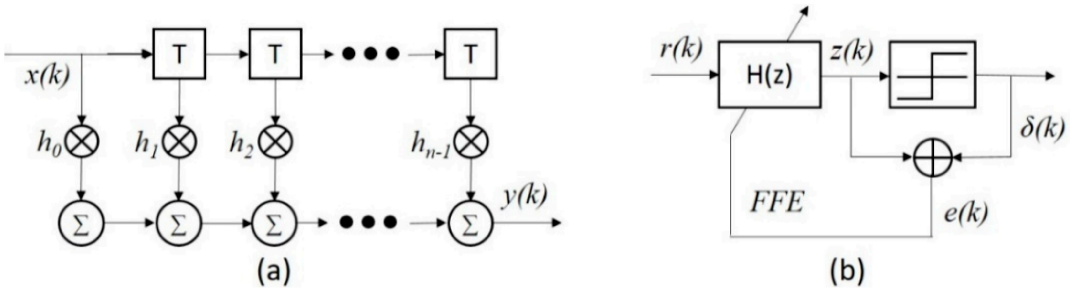


Figure 3.1: (a) Structure of FIR filter; (b) structure of a FFE [63].

an output $y(k)$ on the basis of the following equation [36]:

$$y(k) = \sum_{l=0}^{n-1} h_l x(k-l) \tag{3.1}$$

where k is the sampling instant, $h = [h_0, h_1, h_2, ..., h_{n-1}]$ is the array of the tap weights and N is the number of taps.
In Figure 3.1b, we can see the decision-directed FFE, where the FIR filter is represented by its frequency response $H(z)$. In order to update the tap weights, it is possible to apply an algorithm such as the Zero Forcing (ZF) algorithm, the least mean square algorithm Least Mean Square algorithm (LMS), the recursive least square algorithm Recursive Least Square algorithm (RLS), etc. [35]. The difference between these algorithms consists in the tradeoff between convergence, speed and residual error variance.
For the sake of this section, we only describe the direct decision LMS algorithm. Given the iteration $(k+1)$, the filter tap weights are updated as follows [20]:

$$h(k+1) = h(k) + 2\mu e(k) R(k) \tag{3.2}$$

where μ is the step size, $e(k) = \delta(k) - z(k)$ is the error between the target signal $\delta(k)$ and the output signal $z(k)$. $R(k) = [r(k), r(k-1), r(k-2), ..., r(k-n+1)]$ is the input signal.
The FFE can produce an amplification of high-frequency components that experience losses because of the limited bandwidth of the system. The FFE can perform its operation at the symbol-level or at a higher rate sampling up to the sample-level. They are called symbol-spaced FFE and fractionally-space FFE respectively. The latter is sampled at a multiple of the symbol-rate; this enables the digital implementation of the matched filter and allows the phase recovery

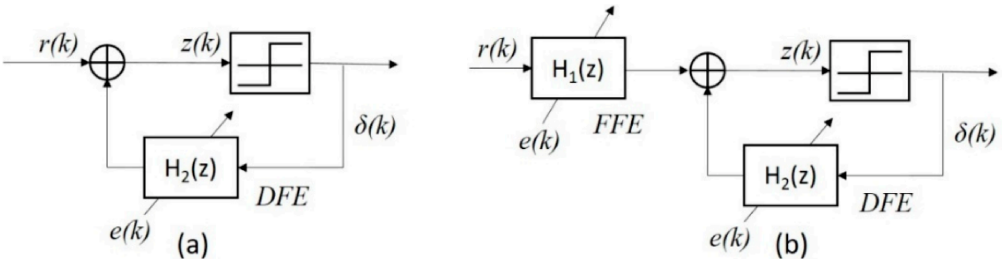


Figure 3.2: (a) DFE structure; (b) FFE and DFE combined [63].

of the received signal. Moreover, it reduces the residual error with the drawback of a higher computational complexity.
As we discussed in Chapter 2, the interplay of Chromatic Dispersion and power-fading has a severe impact on the received signal with the appearance of frequency notches for which the FFE is barely able to compensate. In alternative, it is possible to use a Decision Feedback equalizer (DFE) that is more efficient at mitigating such phenomenon. While the FFE can operate at a sampling rate higher than the symbol rate, the DFE usually takes one sample per symbol and its input is the signal after the decision (Figure 3.2a).

In the DFE implementation, first, the signal before the decision is computed [35]:

$$z(k) = r(k) - \sum_{l=0}^{n-1} h_l \delta(k-l). \tag{3.3}$$

Then, the filter taps weights are updated as follows [35]:

$$h(k+1) = h(k) + 2\mu e(k) Z(k) \tag{3.4}$$

where $Z(k) = [z(k), z(k-1), z(k-2), ..., z(k-n+1)]$.
Adopting a DFE, the equalization of frequency notches succeeds thanks to the pole insertion. However, the process can be affected by error propagation and the filter may be unstable because of the decision feedback scheme.
Furthermore, DFE can manage only post-cursor ISI, while - in presence of CD - the impulse response has both pre-cursor and post-cursor. For the practical implementation, combining a FFE and a DFE is the best choice, as Figure 3.2b shows. The FFE and DFE can be placed at the transmitter to perform a pre-compensation or at the receiver to do the post-compensation [61].

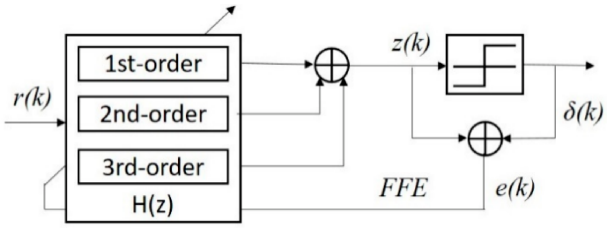


Figure 3.3: Volterra nonlinear equalizer (VNLE) [63].

3.1.2 Volterra Nonlinear Equalizer (VNLE)

FFE and DFE implementation can ensure an efficient compensation of linear impairments. However, there is still a nonlinear distortion caused by the combination of nonlinearity of devices and the DD procedure, resulting in a severe impact on the transmission performance. For this reason, another popular equalization technique is VNLE. In Figure 3.3, we can see its structure whose implementation is based on the previously described FFE. Because of the burden of the computational complexity that this method involves, VNLE is not practical in real applications. To address this issue, different methods have been proposed. One example is the sparse VNLE that can reduce the computational complexity and almost maintain the same performance [12].

3.2 Equalization based on Machine Learning

Recently, ML algorithms have been deployed in optical communications with noteworthy results [64]. Indeed, ML-based equalization methods have proved able to offer a significant improvement in terms of performances. However, sometimes these new methods are more complex than the traditional equalization techniques.
The high complexity is responsible for an increase of the total cost of the equipment, that is a strict constraint in short-reach optical fiber links [63]. Therefore, the goal of the novel ML-based equalization methods is not only to achieve better performances, but also to consider carefully the implementation cost.
In the next section, we will present two widely used ML-based methods. The first involves the use of a Feed Forward Neural Network (FFNN) that is employed instead of an equalizer; the second is a signal processing technique that has become quite popular in communications, i.e. an autoencoder.

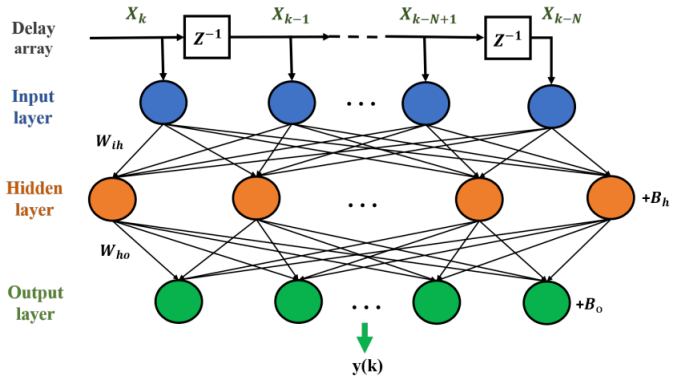


Figure 3.4: Time-delay Feed forward neural network structure [60].

Feed Forward Neural Network (FFNN) Equalizer

FFNNs are widely adopted to provide equalization functionalities in the context of short-reach optical communication systems. The input of a FFNN is a time-delayed series of data. In Figure 3.4 [60], we can see the structure of an FFNN. It has the input layer, the hidden layer and the output layer. While in a common Neural Network (NN), the received data are directly fed to the input layer, in the FFNN, the sampled signals are delayed. Thus, the FFNN is fed with current and past samples to perform the equalization properly. Afterwards, the signal passes through the hidden layer where a nonlinear activation function compensates the nonlinear impairment of the optical channel. As a result, the degraded signal is restored and at the output layer we obtain the equalization coefficients.
The practical drawback of this approach consists in the complexity of the NN that is in conflict with the strict complexity constraints of short-reach optical communications.

Autoencoder

ML technology can enable the joint optimization of the transmitter, the receiver and the communication channel in an end-to-end process. In this context, a communication system with transmitter, receiver and a communication channel in between can be considered as an autoencoder [25].
Typically, an autoencoder is composed by an encode and a decode blocks. The encode block performs a dimensionality reduction on the input data with the purpose of reducing the transmission rate and improving the reliability. Afterwards, the decoding restores the original dimension of the code.

In the specific context of communications, the goal of an autoencoder is to learn how to represent the information data in order to minimize the probability of error at the receiver, augmenting the robustness of the transmitted message with respect to the distortion induced by the channel. While a common autoencoder removes redundancy and compress the input data, an autoencoder in the context of ML for communications adds redundancy and aims at learning an intermediate representation of data that ensures robustness with respect to the channel impairments [32].

3.3 Recent works autoencoder-based focused on PS

In this section, after an introduction to the advent of ML in the context of Optical Communications, we will do a brief recap of some of the most recent papers that investigated the optimization of the PS block. Due to the limited number of works in such direction, we will not consider only works tailored to the short-reach optical link, but we will extend the research to the more general context of Optical Communications.

The advent of Deep Learning in Optical Communications and the widespread adoption of autoencoders

As we discussed, recently, the use of machine learning has become widely used in the context of Optical Communications. We can mention three methods that have been applied to optical channel scenarios: we can refer to the first as a more traditional learning method, then we have the end-to-end learning and the physics-based learning.
The first consists in the replacement of a single signal processing block by a neural network. In this way, it is possible to learn efficient and/or less complex algorithm through data-driven optimization [23].
The second method was presented in [32] in 2017. It proposed a new way to think about communication systems: a transceiver can be designed as an autoencoder, i.e. an end-to-end reconstruction task that aims at jointly optimizing transmitter and receiver components in a single process. This method was adopted for the first time in the context of optical communications by Karanov et al. in 2018 [25]. This is a ”black-box” method that does not allow to easily incorporate existing domain knowledge into the learning process and, moreover,

provides domain solutions that are difficult to interpret.
The third method was presented in [17] in 2020. Its purpose is to find a trade-off between the black box solution provided by an end-to-end process and the potentially sub-optimal approach that applies deep learning techniques to individual signal processing blocks carrying out an individual task [17]. Instead of relying on a generic Neural Network (NN), this methods starts from an existing model and extensively parametrize it. It enables the possibility to initialize the model with clear hyperparameters choices, provides a proper initialization for gradient-based optimizers and allows to interpret the learned solutions providing insight into the problem [17].
Recently, plenty of works have developed new frameworks based on these approaches; among them, we have found several works focused on the design of the pulse shaping block with the use of autoencoders. An autoencoder is a powerful tool that can learn how to communicate efficiently over the optical fiber channel: for instance it can jointly optimize geometric constellations and pulse shaping, while also taking into account linear and nonlinear impairments such as chromatic dispersion and Kerr-nonlinearity.
The first block-based autoencoders were implemented as FFNNs [26].
To further improve the performances and to reduce the transceiver complexity novel solutions were developed.
For instance, Karanov et al. [26] did one of the first works that enabled the potential of end-to-end optimized transmission via deep learning with the use of an end-to-end sliding window Bidirectional Recurrent Neural Network (BRNN) that achieved a significant reduction of the bit-error-rate. In addiction, it enabled higher data rates at a lower computational complexity with respect to systems employing state-of-the-art nonlinear equalizers. The target was a nonlinear channel with memory, such as the short-reach optical channel model that we presented in Chapter 2.
Given these premises, in the next paragraph, we will do a recap of some recent works that propose deep learning techniques in Optical Communications, focusing on the optimization of the pulse shaping block.

Not exhaustive roadmap of recent works on the optimization of the Pulse Shaping block

In [54], Uhlemann et al. proposed a novel design for coherent system. The transmitter implementation involves a trainable symbol mapper and a trainable pulse shaping filter. They proved that the pulse shape filter can be trained to compensate the impairments of the nonlinear optical channel such as chromatic dispersion and Kerr nonlinearity.
In [2], Aoudia and Hoydis presented an end-to-end learning approach to design the pulse shaping and the constellation geometry jointly. This end-to-end optimization of the constellation and of the pulse shaping uses a conventional architecture and, thus, it does not increase the transmitter complexity; tested and validated on the AWGN channel, this technique can be applied to any channel model.
Starting from the framework proposed in [54], Song et al. [45] proposed an autoencoder to perform a joint optimization of the constellation, of the pulse shaping filter and of the modulator to achieve a mitigation of the nonlinearities of the channel. While in [54] it was developed for a single channel, in [45] the target was the design of a superchannel system aimed at minimizing the signal bandwidth and the interchannel interference. Compared to early works, where the autoencoders were implemented as black boxes, in the aforementioned works, the transmitter was split into a concatenation of simpler NNs, each corresponding to one functional block of a conventional communication system and, thus, the NNs parameters can be initialized such that they initially perform close to their conventional counterparts. As a result, the proposed schemes had a better interpretability and reduced training complexity compared to a conventional autoencoder.
Following this direction - that consists in the adoption of the so-called physics-based model - other studies were performed on the individual pulse shaping block.
For instance, in [22], He et al. experimentally verified the concepts introduced in [45], taking advantage of the learned pulse shaping filter in a superchannel system. For the same Digital Signal Processing (DSP) complexity, the proposed method improves the spectral efficiency compared with traditional RRC filters. In addiction, the ML-based pulse shaping filter induces a significant reduction in the number of equalizer taps required, enabling lower DSP complexity and

lower power consumption.
In [55], expanding the linear architectural template of [54], Uhlemann et al. chose the well-known autoencoder as an approach to find an appropriate scalable nonlinear pulseshaping. The proposed framework is realized as a convolutional neural network at both transmitter and receiver. The novelty consists in the introduction of a γ-lifting training procedure tailored to the nonlinear optical fiber channel. As a result, the autoencoder provides a more stable convergence and the design of pulse shape block is achieved. This framework reaches an information rate able to outperform the classic scheme of split digital back propagation at high input powers. In this work, it was considered only a single channel and single polarization scenario, but the idea can be extended to a multi-channel systems or to more complex architectures.

3.4 Summary of recent advancements of interest

We can sum up recent progresses - from the general introduction of deep learning in optical communications to works about the optimization of the individual pulse shaping block - as follows:

- the advent of deep learning in communications [32];
- first application of deep learning to the specific field of optical communications [25];
- the widespread adoption of the autoencoders along with the so-called end-to-end optimization process;
- early studies provided ”black box” solutions and were focused on single-channel implementations;
- more recent studies proposed solutions adopting the so-called physics-based learning [17], implementing the framework as more conventional setups in order to avoid the potential additional complexity that a ”black box” process may imply and also to facilitate the initialization of the parameters and to make the solutions easier to be interpreted; in these new frameworks, the system is made of several blocks implemented as independent neural networks that are optimized jointly (for instance constellation and pulse shaping jointly [2]) in an end-to-end process;
- the specific structure that promotes the joint optimization of the constellation and of the pulse shaping has been proved to have the same computational complexity of more conventional setups [2];

- recent studies extended previous frameworks to superchannel scenarios to provide a better spectral efficiency and to reduce the interchannel interference;
- some studies have focused their attention on the pulse shaping block: the optimization of the pulse shape split between the transmitter and the receiver to enable an end-to-end optimization seems to be a promising research direction; studies have proposed a linear implementation and also a nonlinear implementation of the pulse shape block.

3.5 The idea of this work

In the research papers we analyzed, the optimization of the Pulse Shaping block was aimed at achieving various objectives: enabling high data rates, reducing the Symbol-Error-Rate (SER) and the power consumption, compensating for the channel penalties, reducing the complexity and, for multi-channel scenario, improving the Spectral Efficiency (SE) compared to traditional RRC filters, reducing the interchannel interference, etc.
To sum up, there have been proposed several studies on the Pulse Shaping block that have shown promising directions towards the long-term goal of implementing efficient, reliable and less complex optical communication links.
In this view, in this work we decided to focus our attention on the Pulse Shaping block in order to further investigate the opportunities that the research in this field has enabled and has started to explore.
In particular, we decided to focus on the issues due to the limited bandwidth. Compared to the aforementioned works, implemented with deep learning methods, the proposed framework is much easier and set the less ambitious goal to do a preliminary methodological study on the Pulse Shape block in a scenario with an impaired communication channel. More specifically, we will design a ML-based non-conventional equalization method that we will test and validate on the band-limited AWGN channel, comparing the obtained results with the performances of traditional equalization methods.
In Chapter 5, we will state the purpose of this work presenting a non-conventional setup with ML-based FIR filters that we want to train in order to achieve the equalization task, compensating for the channel non-idealities.

4

Machine Learning applied to Communication Systems

In this Chapter, we will discuss motivations and use cases for ML applied to the context of Digital Communication Systems.

4.1 ML in the context of Digital Communications

On the contrary of fields like image and natural language processing, where ML is already well-established, the application of ML techniques in Communications is a novel research direction. Within this context, there are plenty of aspects that in principle could be solved through Machine Learning techniques: for instance, at the Physical Layer ML could manage baseband signals using channel state information; at the Network Layer ML could be employed for aspects related to the location and traffic loads; finally, at the Application Layer it could compute loads and user preferences [27].
In the next Section, we will explain how the ML approach differs from a traditional engineering design flow. Afterwards, we will do an excursus of the advantages that can provide to Communications, mentioning several use cases at the Physical Layer. Finally, some challenges and future directions will be discussed.

4.1.1 How is it different from the conventional engineering approach?

ML methodology can be considered a promising alternative to the conventional engineering approach for the design of an algorithmic solution. Which are the main differences of the two approaches - the conventional engineering design flow and a novel ML-based methodology? They are represented in Figure 4.1.
Let us start with a brief explanation of a traditional design flow. The acquisition of domain knowledge is the starting point of the conventional flow. The problem of interest is analyzed thoroughly under the assumption of a physic-based model that gives an accurate representation of the set-up under study. Then, the problem is represented by a mathematical model that captures the physics of the problem and that allows the definition of an optimization problem aimed at the design of an algorithm that in principle could achieve optimal performances [44]. The model selection and the design of the algorithm is performed at hand; then the algorithm is implemented and its performance evaluated.
In contrast, providing that it is possible to perform a real experiment in which the system is fed with input data that will be paired with the outputs of the experiment, a Machine Learning based methodology can operate even in the absence of a well-established mathematical model.
It starts with the collection of the examples of the training set from the real experimental setup. This task is potentially easier than the conventional acquisition of domain knowledge, assuming that a sufficiently large number of examples with the desired behaviour is provided. These samples are fed to an algorithm that learns the desired task by the optimization of a performance criterion [43].
Going a step further, it is even possible to include the available domain knowledge in the ML learning process as it can dictate the choice of a specific hypothesis class to use in the training process [44].

4.1.2 Motivations for the use of ML in Digital Communications

In the last decade, ML and its widespread adoption in many fields has started to draw the attention even to the the field of communications thanks to its high

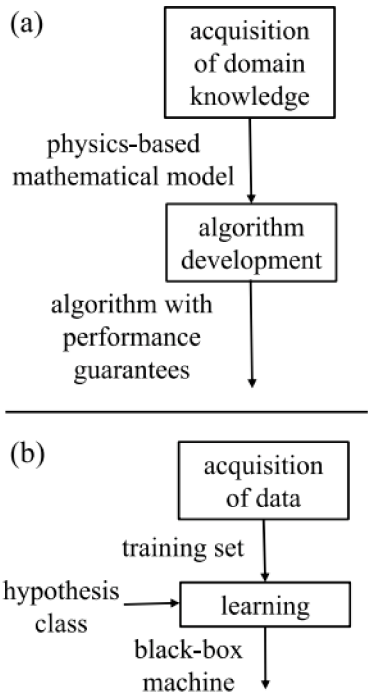


Figure 4.1: (a) Traditional engineering design flow and (b) ML-based methodology

potential that can lead to an improvement of well-established algorithms in terms of reliability, generality, latency, and energy efficiency [32]. Let us analyze more in detail which kind of improvement ML can bring to communications.

System models: Current algorithms in communications need a formal definition of the models to be implemented. A ML communication system does not need a model to represent and transform the information. It can be advantageous because instead of implementing the traditional algorithms that assume ideal system models, ML can offer optimal solutions for the real ones, with imperfections and non-linearities, especially thanks to novel optimization approaches (like the so called end-to-end optimization approach).

Getting rid of functional block structure: Communication systems are modelled as a chain of independent blocks that performs a specific and defined operation. In the design of a communication system, what engineers care about is to ensure the optimization of every block individually. A ML based approach will provide a global view on the optimization task: the system will be trained end-

to-end to achieve the best metrics jointly, considering all components, avoiding a well defined and limiting block structure.

Parallelization: With respect to the conventional "hand-written" algorithms, the use of ML will lead to a high parallelization of processes and, thus, the algorithms obtained from ML will have the potential to be executed faster and at a lower energy cost.

A better exploitation of hardware architectures: The actual algorithms and the use of high-level programming languages are not able to exploit all the potential in terms of computations and energy efficiency that the novel concurrent architectures with distributed memories such as Graphical Processing Unit (GPU) can provide. Thanks to their intrinsic concurrent nature, Neural Networks represent a promising solution to implement efficient algorithms capable of exploiting the parallelism of the hardware architectures. Authors of [32] believe that special purpose code will not be used anymore for algorithms in computing and communications and ML will provide new algorithms trained and optimized for end-to-end loss functions. It will be a versatile approach in which parameters will be readily modified to optimize different tasks exploiting highly concurrent architectures.

Currently, among the subset of Machine Learning, Deep Learning has gained significant attention in the literature, in Wireless Communications [9] and, more recently, even in Optical Communication links [29]. The application of Deep Learning to communication consists in the design of Artificial Neural Networks that can replace or augment DSP blocks (such as equalization and decoding).

Which are the most commonly used methods to perform the learning?

The data-driven optimization strategy can learn new algorithms that can achieve in principle better performances than the traditional DSP blocks in terms of accuracy and computational load.
We can distinguish three different approaches that can be adopted to perform the learning in the context of Communications:

- "Traditional" Learning;
- End-to-end Learning;
- Parametrization of an existing model.

In the "Traditional" Learning, every single block is optimized individually to approximate an input-output function. Whereas, another possible approach consists in the modelling of the entire transceiver design as an end-to-end task in which transmitter and receiver can be optimized jointly.
These two approaches are usually implemented with NNs and, thus, they have the disadvantage that the NN structure is not related to the problem at hand and their high-level operations are difficult to interpret as they are used like "black boxes". On the contrary, a conventional DSP algorithm is based on well-understood mathematical models and relies on a robust theoretical framework. There exists a third approach that does not include the NN implementation: it is possible to start from an existing model and parametrize it; it can be advantageous w.r.t. a conventional NN because hyperparameters can be selected more intuitively and can provide good initialization for gradient-based optimization. As a result, the obtained solutions can be interpreted more clearly, providing significant insight into the problem [5].

4.1.3 Use cases

How can ML provide remarkable solutions that can be applied in the context of Digital Communications?
In this Section, we will analyze more in details several aspects that can be improved in Digital Communications with use of Machine Learning and, more specifically, of Deep Learning. To sum up, we will present two more practical use cases based at the Physical Layer. Even though there already exist optimal solutions for Digital Signal Processing problems at the Physical Layer of Communication Systems, for instance the well-established estimation, detection, and optimization theory, there are also practical problem still without acceptable solutions [4].
Firstly, there are many problems that can rely on a known algorithm to find the optimum, but it is too complex for real-time implementation. Secondly, there are cases where the standard system models are inadequate or incomplete [4].
It is important to underline that it is not possible to completely eliminate errors at the Physical Layer of Communication Systems and this issue is conventionally managed using retransmissions. However, the application of Deep Learning methods can provide an improvement in Communications and give robustness to unexpected behaviours with atypical signals that occasionally occur and that

are not considered and not taken into account in the traditional design flow, represented by an ideal mathematical model that is fed only with typical input signals [4]. On the other hand, the intrinsic fault-tolerance of Physical Layer can turn into a disadvantage because attackers can also exploit atypical signals to perform jamming more efficiently [42].

Use case 1: approximating algorithms Machine Learning - and more precisely Deep learning - can be used to reduce computational complexity of known iterative algorithms that take a long time to reach the convergence [49] in case in which there is a strict latency constraint. There are many optimization problems to be solved in Communication Systems that are prohibitive with real-time applications. A NN can be trained to learn approximately how the solution depends on the input data.
In this stage, the use of domain knowledge for a pre-processing is recommended because in this way the learning process will not start from scratch and rediscover known features of the input data, but it will focus on the actual problem that it is supposed to solve.
Using one of the approaches discussed in the previous Section, whether starting from a "black box" design or using a pre-defined model, the NN is capable of implementing a shortcut-algorithm to achieve a good tradeoff between accuracy and computational complexity. In addiction, they can solve a practical problem related to hardware implementation and its time-to-market, i.e. the time it takes from the algorithmic design to the launch of a new product. In order to implement on hardware a new algorithm, instead of implementing a dedicated circuit based on it, with the help of Deep Learning, it is possible to use a general-purpose circuit that implements a NN. The NN can be trained to perform the required algorithmic task and, finally, the learned parameters can be loaded onto the circuit. As a result, the time-to-market for the hardware implementation can be significantly reduced. However, the generation of desired outputs is highly computationally-demanding: the training can take a lot of time if the algorithm is complex and this means that basically the complexity issue is moved from the algorithmic run time to the design process [4]. Therefore, it is important to be careful and to find a tradeoff between these two aspects in order to exploit the benefit that Machine Learning can provide.

Use case 2: inversion of an unknown function Another important application is the inversion of an unknown function, in particular in the case of a non-linear hardware or channels.
Conventionally, the distortion is inverted as follows: an appropriate parametrized model is identified, then the parameters are estimated from measurements, and finally the inverse function is created. However, this approach is suboptimal and error-prone.
Training a NN to invert the function is an alternative that works without defining a model or estimating the parameters explicitly. It is important to underline that only if suboptimal conventional algorithms exist, a learned algorithm can theoretically provide better performance and robustness (under the assumption that the training is carried out successfully).
This application can be useful for time-varying channels because it can enable online-learning by sending occasionally predefined reference signals to generate new training data.
To sum up, the aim of a successful utilization of Machine Learning is to identify tasks in Communication Systems that currently lack an optimal solution because in these specific cases it actually represents an opportunity to go beyond the current state-of-art [4].

4.1.4 Future directions and open research challenges

On the contrary of other fields in which ML is pervasive and well-established, the context of Communications still lacks open datasets and benchmarks to compare the performance of ML models and algorithms.
Therefore, it is desirable to develop standardized datasets and a set of problems to allow the possibility to compare novel algorithms.
Which are the aspects that need to be considered and defined in the context of ML for Communications?

Development of benchmarks One of the most important challenge in the ML-based communications is the development of benchmarks and open datasets. In other fields like computer vision or voice recognition, there already exist common datasets like MNIST7 or ImageNet8 for this purpose [32]. One main difference with other fields is that, in Communications, the signals are generated synthetically and, thus, establishing a set of common problems and associated

datasets could be done with standard routines. As a result, researchers could easily benchmark and compare the developed new models and algorithms.

How to represent data, choose the loss-function and implement the training? The binary signals that are injected into the communication channel have been commonly represented as one-hot vectors, complex symbols or integers [32].
The optimal representation depends on the model architecture, on the learning objective and on the loss function. In addiction, another issues that raises in this context is at which SNR values the training should be performed. The system is supposed to operate over any SNR and, thus, the results should be tested and generalized to arbitrary SNR. However, if the test is performed at certain SNR ranges, it does not necessarily imply that the test will succeed at different ranges. If the training is performed at low SNR, we are still blind on high SNRs scenarios. As a possible suggestion to deal with this generalization issue, in [15] it has been observed that if the training is started at high SNR and then it is gradually lowered with the epochs, the performance improved for that specific application.
Furthermore, it is important to make a proper and careful choice of the loss function that is very model-dependent. For instance, a typical choice for a classification problem is the Cross-Entropy. In general, the choice is not obvious, especially for such a novel field. A bad choice of the loss function can turn into poor results [32]. Even other practical choices such as the model architecture and the parameters of the training for the application of the Stochastic Gradient Descent (SGD) algorithm (e.g. the learning rate) have to be evaluated thoroughly. There exist some guidelines [43] that can be followed, but the research in this field is still in progress for other important aspects such as the hyperparameters selection.

5

Problem statement

Starting from the issues experienced in the real scenario of the short-reach optical link (see Chapter 2) and from the ideas we came up with after the Literature Review (see Chapter 3), in this work, we decided to evaluate a Machine Learning-based non-conventional setup with the long-term aim to provide a possible solution to complex nonlinear scenarios that can not be solved analytically such as the optical channel that we analyzed.
Because of the inexperience and limited amount of time, we tested and validated the proposed method only on the band-limited AWGN channel. In future works, this method can be extended to more complex channel like the nonlinear short-reach optical channel that motivated and gave rise to this work.
Therefore, in this work, we placed ourselves at a preliminary stage in which the query is still at the methodological level. The method that we propose consists in the development of a system with non-conventional Pulse Shape filter and Matched Filter implemented as trainable FIRs provided with a Machine Learning framework. The goal of the ML framework is to train the FIR taps to ensure an optimal performance and to compensate for the channel impairments that degrade the signal along the propagation. Assuming a time-varying nonlinear channel, such as the short-reach link optical channel, the long-term objective of this method is to enable the equalization of the channel with an online Machine Learning process implemented with low complexity and able to adapt and to respond to the time variation and to the nonlinearities of the channel. In other words, we want to ensure an ISI-free transmission in a nonlinear and noisy channel relying on an end-to-end learning process that involves a Pulse Shape filter

at the transmitter and a Matched Filter at the receiver, that are hence optimized jointly.
On this path, in this work we have a short-term and less ambitious purpose: our focus is on the Pulse Shaping block of the communication system and our goal is to test and validate the proposed ML-based method.
To achieve this goal, we decided to simulate our ML-based structure on the band-limited AWGN channel whose response is well-known and, thus, can be inverted analytically to achieve the channel equalization. As a result, it allowed us to increase the interpretability of the obtained results and to easily provide a benchmark to achieve our final goal, the validation of the proposed method.
In the view of extending the proposed framework to more complex channels like the short-reach optical link that inspired this work, the validation is an essential task that has to be completed in order to make our framework a reliable structure, potentially suitable to any type of channel. In general, without the validation step, we would not be able to extend a framework to further experiments on problems that still lack a solution.
To sum up, with our preliminary analysis, we tested and validated a non-conventional ML-based equalization method on the band-limited AWGN channel. In this way, we started exploring the opportunities that the standalone Pulse Shape block can enable and seeing whether it can be promising for further investigations on scenarios with a more complex channel.

5.1 THE SYSTEM MODEL

In Figure 5.1, we can see the block diagram of the considered system model. As we said, in this work, we simulated only the band-limited AWGN channel, that is a linear and static model. However, the problem statement involves an arbitrary time-variant noisy channel that may be distorted by both linear and nonlinear effects. As we saw in Chapter 4, ML can be a powerful tool in context where the nonlinearity makes the simple inversion of the channel infeasible or even in case in which there is not well-defined channel model to rely on.
In order to compensate for the channel impairments, we propose three different equalization setups.

Scenario 1): learning the parameters at the transmitter. In the first scenario, the system is provided with an adaptive trainable FIR PS and with a RRC at

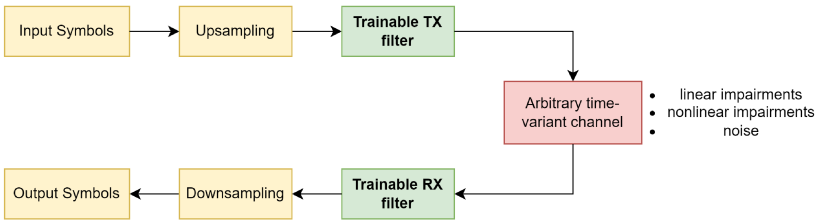


Figure 5.1: Scenario with a noisy channel affected by both linear and nonlinear effects with a trainable FIR filter at the Transmitter and/or at the Receiver.

the receiver. Thus, the goal is to perform an online training of the FIR in order to adapt it to the time variation of the channel. Ideally, the proposed PS should be able to provide two functionalities: it should act in behalf of a conventional Nyquist Pulse Shaper and of a pre-distortion block that compensate for the channel impairments. Therefore, the Pulse Shaper that we are looking for is an unknown Pulse Shape that could provide an ISI-free transmission along an arbitrary complex channel.

Scenario 2): learning the parameters at the Receiver. Similarly, we can define the second scenario. It involves a RRC at the transmitter and a trainable FIR filter at the receiver. The purpose is that to ensure an online training of the FIR and adapt it to the time variation of a nonlinear channel. Therefore, compared to a conventional setup, the desired FIR should provide the functionalities of a Matched Filter and of a Post-Equalizer.

Scenario 3): learning the parameters at the transmitter and at the receiver jointly. Finally, we have the third scenario in which we propose a quite novel setup that has been widely used in the communications research field in the last few years, i.e. an end-to-end optimization. As we discussed in Chapter 4, this novel approach consists in the joint optimization of several blocks in order to overcome the conventional method where every block is optimized individually. In this work, the blocks that we want to optimize jointly are a trainable FIR filter at the transmitter and a trainable FIR filter at the receiver to mitigate the channel impairments and to provide an ISI-free transmission.

5.2 Performance metrics

In order to evaluate the results of the simulations, we divide the metrics we used in this work into quantitative and qualitative metrics.

5.2.1 Quantitative metrics

Mean Squared Error (MSE) We used the MSE as a loss function $\mathcal{L}(\cdot)$ in the training of the ML model. It is defined as the average of the squared differences between predicted and true values. Given N, the number of the FIR parameters to be optimized we can define the loss function as follows:

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2 = \mathbb{E}[(y - \hat{y})^2] \tag{5.1}$$

where y_k is the target, $\hat{y}_k$ is the predicted value.

Energy per bit to noise power spectral density ratio. Communication systems are designed to compensate for distortions at the receiver and this capability depends on the tradeoff between the power of the signal and the power of the noise. The relation between the signal and the noise is called the Signal-to-Noise Ratio (SNR):

$$SNR = \frac{P_{signal}}{P_{noise}}, \tag{5.2}$$

where P_{signal} is the average power of the transmitted signal and P_{noise} is the average power of the noise. Since it is the ratio between two powers, the SNR is a dimensionless quantity. In our framework, instead of adopting the SNR, we used the energy per symbol to noise power spectral density ratio, that is a normalized SNR, more suitable to the context of digital communications in which we operate. It is expressed as:

$$\frac{E_s}{N_0} \tag{5.3}$$

where E_s is the average symbol energy that considers both In-Phase (I) and Quadrature (Q) samples; N_0 is the average Power Spectral Density (PSD) of the noise [11].

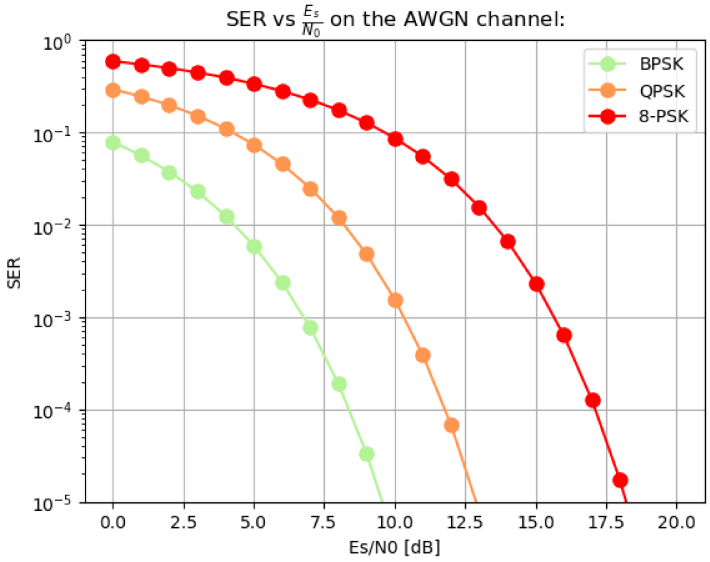


Figure 5.2: Symbol-Error Rate VS $\frac{E_s}{N_0}$ on the AWGN channel.

Symbol-Error Rate (SER) Because of noise, interference and distortions, bits synchronization errors that occur during a digital transmission over a communication channel, some symbols of the transmitted sequence are altered and this results into a symbol-error probability P_e. The SER curves are plotted as $\log_{10} P_e$ versus the Energy per Symbol to Noise Power Spectral Density Ratio in dB, i.e., $10 \log_{10}(\frac{E_s}{N_0})$, resulting in a waterfall-like shape as we can notice in Figure 5.2 that shows the theoretical SER curves for BPSK, QPSK and 8-PSK modulation formats.

5.2.2 Qualitative metrics

IQ or constellation diagrams An IQ diagram is a representation of a signal modulated by a digital modulation scheme such as MQAM or MPSK. The signal is represented in the complex plane as a xy-plane scatter diagram at symbol sampling instants. The x axis shows the In-Phase component and the y axis represents the Quadrature component. In Figure 5.3, we can see the IQ diagrams for 8-PSK modulation format at SNR = 20 dB.

Eye-diagrams An eye diagram is widely used in the field of digital communication in order to exhibit the quality and the characteristics of a signal.
For example, it is useful for the evaluation of the impact that noise and other effects such as ISI generate on the transmitted signal along the propagation as

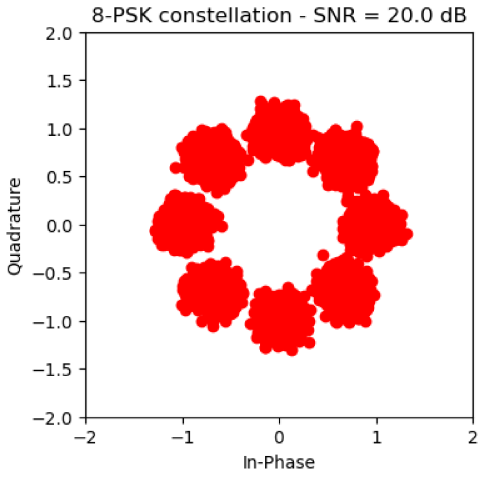


Figure 5.3: Constellation diagram for 8-PSK modulation format.

Figure 5.4 suggests. The eye-diagram is obtained by iteratively taking (usually) two symbols of the signal of interest and by plotting them on the same figure. The resulting figure consists in the superposition of the considered sequence of symbols and can provide useful information about the quality of the signal. Since we are dealing with complex waveforms, we consider an eye-diagram for both the real part and the imaginary part of the signal.

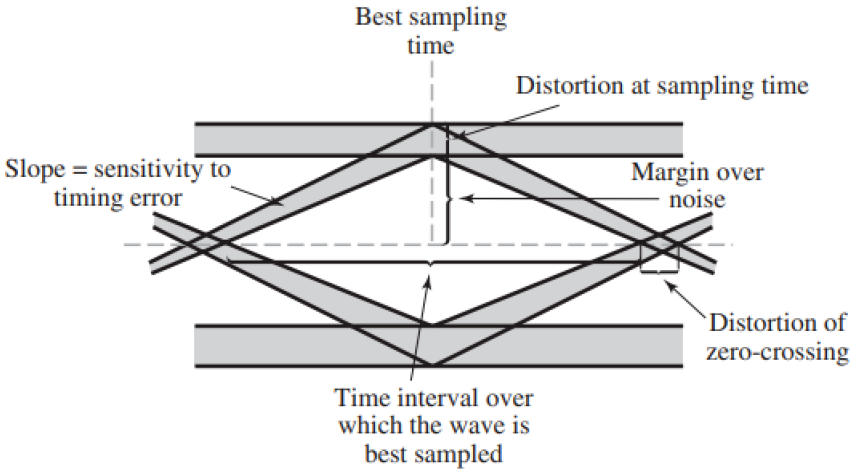


Figure 5.4: Interpretation of the eye pattern for a baseband binary data transmission system [19].

6

Machine Learning: theoretical background

After a brief introduction to Machine Learning, in this Chapter, we will focus on the subset of Supervised Learning that we used in this work. First, we will present the ML process in the context of Supervised Learning and then we will present specific tools that we adopted in this work, such as optimizers and optimization algorithms.

6.1 Introduction to ML

Machine Learning (ML) can be defined as the ability of Artificial Intelligence systems to acquire knowledge capturing patterns from data [31]. The unprecedented availability of data and computing resources have shed light on such data-driven ML techniques on a vast number of engineering field, from speech to imagine analysis [16] to communications [23].
ML theory is rooted in classical linear algebra and probability [7], more specifically it relies on decades old algorithms - for instance backpropagation [21] - and on other mathematical tools such as regularization techniques and adaptive learning rate schedules [44].
In the next Section, we will provide a classification of ML methods.

6.1.1 Taxonomy of Machine Learning Methods

ML methods can be divided into three main classes, depending on how the data are fed into the system.

Supervised learning: In supervised learning, the a training set - the input samples along with their own labels - is fed to the the learning process and, thus, the ML is applied in order to learn the mapping between input and output spaces. We can distinguish into two different tasks in Supervised Learning: in the first instance, there is the regression problem, where the prediction is a continuous variable, and the second is the classification task, where the input associated to a discrete predefined output [27].

Unsupervised learning: In unsupervised learning, the input data are not labelled, thus the training set consists of inputs without any assigned output. Therefore, the goal of unsupervised learning is that of find patterns and cluster together data, hence assigning a label to each input point.

Reinforcement learning: Reinforcement learning lies in between supervised and unsupervised learning. After selecting an output for a given input or observation, a feedback from the environment is received by the reinforcement learning algorithm that expresses the degree to which the output fulfils the goals of the learner. Reinforcement learning requires a more complex analytical framework that takes advantage of Markov Decision Processes [51].

Among the three presented classes, Supervised learning is the most common type of learning, has a robust theoretical basis [56] and relies on well-established algorithmic tools [44]. In the next sections, we will focus on Supervised Learning, that will be the subset of Machine Learning that we will use in this work.

6.2 The Supervised Learning process

In this Section, we will define the Supervised Learning process. Given a training set $\mathcal{D}$ of cardinality N where (x_n, y_n) are the input-output pairs with $n = 1, ..., N$, we have a Supervised Learning problem in which we want to predict the output y for an unobserved input x, thus an input that is not present in

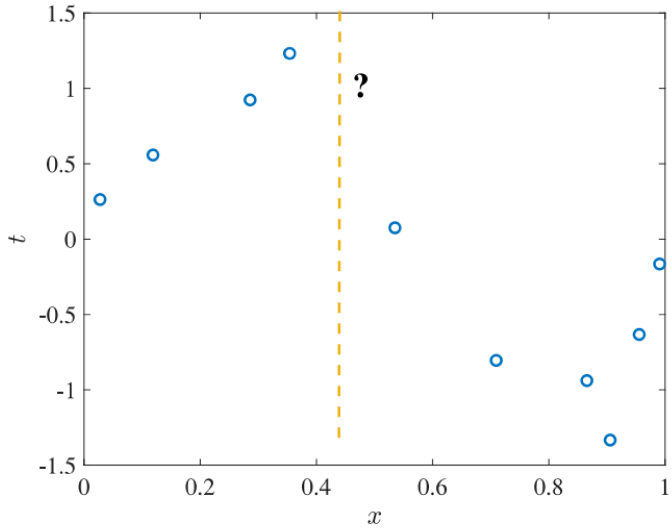


Figure 6.1: Regression problem (Supervised Learning): (x_n, y_n) are the input-output training examples, with $n = 1, ..., N$ and "?" is the unobserved value that we want to predict.

the training set $\mathcal{D}$. In other words, it aims at deriving a predictor $\hat{y}(x)$ to generalize the mapping in $\mathcal{D}$ to input x that does not belong to the training set D.
This generalization property is what distinguishes the *learning* from the *memorizing*: the former allows to extend the capability of classifying any input x that belongs to the relevant input space, while the latter relies only on recorded input x_n in the training set to produce the output y_n.
In order to derive a predictor given the training set $\mathcal{D}$ with all its associated labels, it is necessary to make assumptions about the input-output mapping.
Formally, this concept is expressed by the no free-lunch theorem [56]. It states that the available observations can not be generalized and, thus, can not be extended to input data that are outside the training set $\mathcal{D}$ unless there are some preliminary assumptions.
Therefore, some assumptions about the problem have to be made; they are called the inductive bias. For instance, a hypothetical inductive bias could be the assumption that the mapping is a polynomial function of a certain order [44]; as an example, this is the assumption that we could make about Figure 6.1, where a regression problem is represented.

6.2.1 Formal definition of Supervised Learning

The first assumption is that in the training set $\mathcal{D}$ we have input-output pairs generated as $(x_n, y_n) \sim p(x, y)$, with $n = 1, ..., N$. In other words, each pair (x_n, y_n) is an independent and identically distributed sample that comes from the same joint distribution $p(x, y)$.
The predictor $\hat{y}$ that we are seeking should perform well on any possible relevant input x and, thus, any test pair $(x, y) \sim p(x, y)$ independently generated from the pairs of the training set $\mathcal{D}$ should be accurately classified by the predictor $\hat{y}$ [44].
Given a test sample (x, y), the prediction is evaluated by a loss function $\mathcal{L}(y, \hat{y}(x))$.
One of the most commonly used loss functions is the quadratic loss $\mathcal{L}(y, \hat{y}(x)) = (y - \hat{y})^2$ for regression problems. While, for classification problems, a common choice is the error rate $\mathcal{L}(y, \hat{y}) = 1(y \neq \hat{y})$ that returns a 0 is the sample is correctly classified, 1 otherwise.
The learning process is aimed at minimizing the so-called generalization loss, that is the average loss on the test set. Assuming the popular frequentist approach [43], given a predictor $\hat{y}$, we can formally define the generalization loss as follows:

$$\mathcal{L}_p(\hat{y}) = E_{(x,y)\sim p(x,y)}[\mathcal{L}(y, \hat{y}(x)]. \tag{6.1}$$

6.2.2 Model selection, Learning and Inference

Assuming that there is no domain knowledge and the distribution $p(x, y)$ is not known, in order to solve the learning problem, we need to find a predictor that minimizes the generalization loss $\mathcal{L}_p(\hat{y})$.
The Machine Learning process can be divided into the three following phases:

1. Model selection (inductive bias): The first phase consists in the definition of a specific class of hypotheses, that represents the so-called model. The selection of the hypothesis class is a pre-requisite for learning. Assuming a probabilistic framework, the hypothesis class, or model, consists in a set of probability distributions parametrized by a vector θ. In Supervised Learning, the model can be:

- Generative: when it specifies a family of joint distributions $p(x, y|\theta)$; the structure that is present in the data can be captured in a more accurate way and, as a result, the performance of the predictor can be improved [6];

- Discriminative: when it parametrizes directly the predictive distribution as $p(y|x, \theta)$; it means that no assumptions are made about the distribution of the input x and, thus, it is less likely to be biased.

In both cases, the hypothesis class is selected among a set of probability distributions that results into efficient learning algorithms in Phase 2.

2. Learning: The second phase consists in the optimization of a learning criterion with the aim of finding a parameter vector θ and identifying a distribution $p(x, y|\theta)$ or $p(y|x, \theta)$, if we are considering a generative or a discriminative model respectively.

3. Inference: At this third phase, in the inference step, the predictor $\hat{y}(x)$ is obtained from the learned model:

$$\hat{y}(x) = \arg\min_{\hat{y}} E_{y \sim p(y|x)}[\mathcal{L}(y, \hat{y})|x]. \tag{6.2}$$

The test data on which the predictor $\hat{y}$ is evaluated have to be different from the training set $\mathcal{D}$. In the next Section, the learning process will be analyzed more in detail and also the model selection will be explained thoroughly [44].

Learning:

The goal of learning is to obtain a predictor able to minimize the generalization error. However, since we do not know the true distribution $p(x, y)$, this task in not achievable.
Thus, we have to rely on the training set $\mathcal{D}$ to have an alternative learning criterion. Having a probabilistic model, a commonly used learning criterion is Maximum Likelihood (ML). This criterion allows the selection of a value of θ in the parametrized set of models $p(x, y|\theta)$ or $p(y|x, \theta)$. The problem is solved maximizing the log-likelihood function:

$$\text{maximize } \ln p(\mathcal{D}|\theta) \tag{6.3}$$

where $p(\mathcal{D}|\theta)$ is the probability of the dataset $\mathcal{D}$ for a given value of θ. Assuming that we are using a discriminative model and that data points in D are i.i.d.,

we can rewrite the log-likelihood as follows:

$$\ln p(D|\theta) = \sum_{n=1}^{N} = \ln p(y_n|x_n, \theta). \tag{6.4}$$

There is rarely an analytical solution and typically Stochastic Gradient Descent is used to solve it.
At each iteration, the mini-batches, i.e. subsets of examples, are selected from the training set and the parameters θ are updated by a quantity proportional to the gradient of the log-likelihood function. The learning rule is the following:

$$\theta \leftarrow \theta + \gamma \nabla p(y_n|x_n, \theta) \tag{6.5}$$

where $\gamma > 0$ is the learning rate.
In the context of neural networks, the gradient is computed with the backpropagation algorithm [16], [43]. The learning algorithm is defined by several hyperparameters; one of them is the learning rate. In order to improve the ML process, regularization techniques can be used [16], [43]. During the learning, the selection of the model can be enhanced via validation.
When regularization is performed, a penalty term θ-dependent has to be added to the log-likelihood. In this way, it is possible to avoid that the learned parameters θ assume values that are a priori unlikely and that could be a possible index that overfitting has occurred.

Model selection

The selection of the model consists in the definition of the inductive bias used in the learning process. There are plenty of aspects that are considered within this stage: the model order selection, the inclusion of domain knowledge and the tuning of the hyperparameters of the learning algorithm.
In this Section, we will consider only the model order selection as an example of the model selection phase.
Thus, given a set of models with different order of complexity, we want to select the model order that will be used during the learning phase.
Let us analyze a regression problem whose output y is a polynomial function of order M of the input x plus zero-mean and standard deviation equal to 1

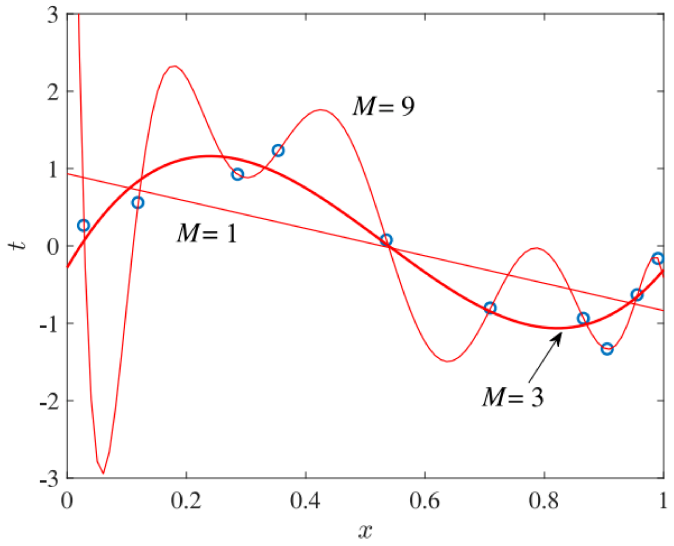


Figure 6.2: Training set and a predictor trained for different values of the model order M.

Gaussian noise:

$$\sum_{m=0}^{M-1} w_m x^m + \mathcal{N}(0,1). \tag{6.6}$$

The parameters θ are given by the weights $w = [w_0, ..., w_{M-1}]^T$. To sum up, in the Model Selection the model order M is chosen; afterwards, in the learning process, the weights w are learned and, eventually, the optimal predictor can be obtained during the Inference. Assuming quadratic loss, the optimal predictor is $\hat{y}(x) = \sum_{m=0}^{M-1} w_m x^m$ for the learned parameters w. In Figure 6.2, we can see a training sample and some possible functions that fit the data with a different M polynomial order. Observing the Figure, we can notice that when $M = 1$, the predictor $\hat{y}(x)$ is not able to capture the mapping of the training data, or in order words, it underfits them, resulting in a large training loss, that quantifies the error between the points in the training set and the predictor defined by the learned weights w. The training loss will be:

$$\mathcal{L}_D(w) = \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y_n}(x_n))^2. \tag{6.7}$$

On the order hand, with $M = 9$, the predictor fits so well the training data that it overfits them. Therefore, the model is too complex and does not perform well outside the training data, giving inaccurate predictions. It is necessary to find

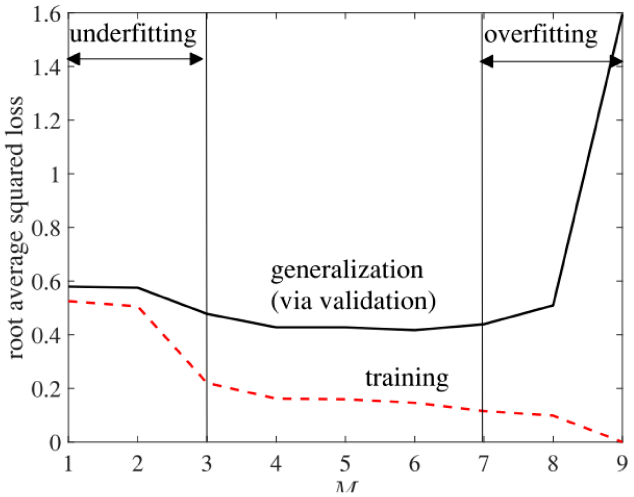


Figure 6.3: Training loss and generalization loss that resulted from the validation, as a function of the model order M represented in Figure 6.2.

a tradeoff between underfitting and overfitting, choosing a model order in between (for instance $M = 3$). This discussion implies that the underfitting can be detected by observing the training data $\mathcal{D}$ evaluating the training loss. Whereas, to detect the overfitting we need to check how the predictor performs outside the training data $\mathcal{D}$.

As a result, the Model Selection phase relies not only on the observation of the training set, but - in order to generalize the performance of the predictor - also other data needs to be considered. This procedure is called validation. It can be performed dividing the available data into two sets, the training set $\mathcal{D}$, that will be used for the learning, and the validation set, that allows to compute an estimate of the generalization loss. Other form of validation can be adopted, for instance the more sophisticated cross-validation [51].

In Figure 6.3, we can see the training loss and also the generalization loss resulted from the validation. When M is too large, the validation loss starts diverging from the training loss and thus the obtained predictor overfits the training data. While, when M is too small, underfitting occurs and can be detected by just observing the training loss as we said.

To sum up, validation is a fundamental phase in the selection of all the parameters used by the learning algorithm and of the hyperparameters.

Finally, after that the training and the validation have been completed, the performance of the selected model needs to be estimated with the use of a different data set, the so-called test set. The reason is that the generalization loss obtained

from the validation is a biased estimate of the true generalization loss because of the Model Selection phase. The test set do not have to be used during the learning process and should only be used once in order to test the trained predictor [44].

6.3 Optimization task

Up to now, we have provided a description of the learning process, that consists in adjusting the parameters θ in order to minimize the loss function $\mathcal{L}$. The problem to find the best choice for the parameters θ is an optimization problem and can be solved with the use of optimization algorithms. To achieve the learning task, there are several types of optimizers that can be used by the model, from gradient-based algorithms to derivative-free optimizers. In the next Section, after a brief introduction about optimizers and how to classify them, we will explore more in details the algorithms used in this work.

6.3.1 Optimizers

Most Machine Learning problems can be solved as optimization problems. The optimization task has to deal with different difficulties and challenges. Optimizers are the algorithms that are used to minimize or maximize an objective function, the so called loss function $\mathcal{L}(\cdot)$. During the training process, they adjust the model parameters with the aim of finding the optimal set of parameters that result in the best performance of the Machine Learning model on the given task.
The application of the Machine Learning models relies on the effectiveness of the numerical optimization algorithms. The development of the optimization methods is hence a crucial aspects in Machine Learning to achieve good performances. On the basis of the gradient knowledge, it is possible to divide optimization methods into three categories [50]:

- first-order optimization methods such as the stochastic gradient method and its variants;
- high-order optimization methods, such as the Newton's method;
- heuristic derivative-free optimization methods, for instance the coordinate descent method.

The difference between first-order and high-order optimization methods consists in the fact that the former converges at a lower speed than the latter, but they deal with less computational complexity.
The first-order and the high-order algorithms are both gradient-based optimization algorithms as they rely on the availability of the gradient to be executed. The gradient has be computed with a proper algorithm, for instance with the well-known Backpropagation.
In some some cases, the derivative of the objective function is difficult to compute or it does not exist. Therefore, instead of using a gradient-based methods, derivative-free optimization methods are applied. These methods can be implemented with a heuristic search based on empirical rules or they can fit the objective function with samples. Derivative-free optimization methods can also work in conjunction with gradient-based methods.
The choice of the optimizer has to be performed carefully because it can have a significant impact on the training process, convergence speed, and final model performance [50]. For this reason, in practice, different optimizers and learning rates are explored in order to find the best combination for a particular problem.
On this path, in this work, we will take advantage of first-order optimization methods - SGD and Adam Optimizers - and even of Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm, a derivative-free optimization method [50].

6.4 Gradient-based techniques:

In this Section we will introduce the Backpropagation, Gradient Descent algorithm and two of its variants that we used in this work. Related to Backpropagation, it is important to underline that in the context of Supervised Learning as we are, the Backpropagation algorithm is not a common choice as it is typically used for training neural networks with multiple hidden layers. However, in view of expanding this framework to more complex and realistic channels, we decided to adopt it anyway in our linear regression model.

6.4.1 Backpropagation: computation of the Gradient

The Backpropagation algorithm [40], [41] is a widely used technique in Deep Learning as it allows to compute the gradient with a simple and inexpensive procedure, overcoming the computational complexity that a numerical evaluation of the gradient could imply. Backpropagation algorithm is commonly used within multi-layer Neural Networks, but in principle it can compute derivatives for any function (providing that it is possible to define them) [16]. As we outlined, Backpropagation is usually coupled with a learning algorithm, such as SGD, to perform the learning task using the computed gradient.
In learning algorithms, the Gradient that is needed is the Gradient of the loss function w.r.t. to the parameters θ to be learned. In addiction, there could be other Machine Learning tasks that require the computation of the derivatives and the Backpropagation algorithm can be applied as well. The main idea of the Backpropagation algorithm consists in the propagation of the computed derivatives through a network (and for this reason it is used to train NNs), avoiding redundant and unnecessary calculations [16].

6.4.2 Gradient Descent

In Machine Learning, the most commonly used optimizers are based on Gradient Descent. Gradient Descent method is an iterative procedure that consists in the update of the parameters to be learned in the opposite direction of the gradient of the objective function [50]. As a result, the parameters vary and gradually lead to an optimal value for the objective function. At each iteration the Gradient is multiplied by the learning rate η and determines the step size on which depends the number of iterations needed to reach the convergence [39].
Let us provide a formal expression of the Gradient Descent method. Given a linear regression model, we have that $f(x) = \hat{y}$ is the function to be learned, $\mathcal{L}(\theta)$ is the loss function and θ the parameters that we want to optimize. Applying Gradient Descent allows to minimize the loss function:

$$\mathcal{L}(\theta) = \frac{1}{2N} \sum_{i=1}^{N} (y^i - f(x^i))^2 \tag{6.8}$$

where $f(x) = \sum_{j=1}^{D} \theta_j x_j$ is the estimated output, N is the number of training samples, D the number of input features, x^i is an independent variable with

$x^i = (x_1^i, x_2^i, ..., x_D^i)$ for $i = 1, ..., N$ and y^i is the target output.
The Gradient Descent procedure involves the two following steps:

1. Compute the partial derivative of $\mathcal{L}(\theta)$ w.r.t. to each θ_j in order to get the gradient:
$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta_j} = -\frac{1}{N}\sum_{i=1}^{N}(y^i - f(x^i))x_j^i. \tag{6.9}$$
2. Update each θ_j in the opposite direction of the Gradient in order to minimize the loss function:
$$\theta_j' = \theta_j + \eta \cdot \frac{1}{N}\sum_{i=1}^{N}(y^i - f(x^i))x_j^i. \tag{6.10}$$

The implementation of Gradient Descent is quite simple. When the objective function is convex, we obtain the global optimal. Since in each iteration step all the training data are used, this method is also called the batch Gradient Descent. Given N samples and a D-dimensional independent variable x, the computation complexity is $O(ND)$ at each iteration. It is a high computational cost, especially if we deal with large-scale data. In order to overcome the complexity issue, other methods were developed such as the SGD.

6.4.3 A Gradient Descent Variant: Stochastic Gradient Descent

SGD [38] is more lightweight than Gradient Descent in terms of computation and, thus, it allows an online update of the parameters.
Instead of directly calculating the Gradient, at each iteration, Stochastic Gradient Descent selects one sample i randomly and afterwards - on the basis of such sample - it updates the Gradient. Even though the stochastic Gradient does not use all the samples to compute the Gradient, it is an unbiased estimate of the true Gradient [38]. SGD is advantageous as its cost is independent from the number of samples and it can reduce the update time and thus, accelerate significantly the calculations, reducing some computational redundancy [50].
In Gradient Descent, the loss function was computed considering all the samples x_i with $i = 1, ..., N$ as we saw in Equation 6.8. Whereas, in Stochastic Gradient Descent, a random sample i is selected and the the loss function will be $\mathcal{L}^*(\theta)$:

$$\mathcal{L}^*(\theta) = \frac{1}{2}(y^i - \hat{y}^i)^2. \tag{6.11}$$

As a result, the Gradient update will be reduced to:

$$\theta' = \theta + \eta(y^i - \hat{y}^i)x^i. \tag{6.12}$$

The computation complexity for each iteration is $O(D)$ where D is the number of features. On the one hand, using SGD allows a much faster update rate than that of Gradient Descent applied to a very large number N of samples. On the other hand, more iterations will be necessary to reach the convergence. Nevertheless, given the small cost of each iteration, the SGD is still able to reduce the computational complexity and to accelerate the convergence, achieving a significant improvement in terms of efficiency w.r.t. the Gradient Descent [50].
However, SGD has a drawback: the random selection of the sample i introduces some noise in the process and, thus, the Gradient direction will oscillate because the search process proceeds blindly in the solution space.
In Gradient Descent, the update of the parameters is always directed to the optimal value. Whereas, the direction of the update in SGD is biased. In order to find a tradeoff between the two methods, the Mini-batch Gradient Descent (MSGD) method was developed [38]. To address the problem of the high variance of the Gradients and the instability of SGD algorithm, at each iteration MSGD uses b independent identically distributed samples (usually a value between 50 and 256 [39]) to update the parameters.
In the deterministic batch Gradient Descent, it may happen that the algorithm generates an outcome that is just a local minimum of the objective function. Whereas, in SGD - because of its stochastic nature - it is more likely to find a global optimum solution as the algorithm explores different minimum thanks to the fluctuations of the algorithm. On the other hand, these fluctuations may decelerate the convergence of SGD.
Another crucial aspect to be considered in SGD is the choice of the learning rate [39]. If it is too small, the convergence will occur in a longer time. Even a too large learning rate is an obstacle to reach the convergence as it would cause the fluctuation of the loss function around a minimum. To address this problem, it is possible to set a list of learning rates to be tested and tune the learning rate online, during the learning process [8]. However, the list has to be defined in advance on the basis of the specific dataset. In any case, this solution provides only one learning rate for all the parameters and, on the contrary, they may need different rates to to be updated properly [10].

In addiction, another challenge is to allow the objective function to escape from the trap of an infinite number of local minimum or from "saddle points" [13]. This latter are quite insidious because they have a positive slope in one direction and a negative in the other direction and gradient values in all directions are zero.

6.4.4 SGD improvements: Adaptive Learning Rate methods

As we explained, in SGD the learning rate has to be regularized "at hand". It a very important aspect because it considerably influences the effectiveness of the SGD. For this reason, adaptive methods aimed at adjusting the learning rate automatically have been proposed. They are characterized by the fact that they do not need to adjust parameters and converge quite fast, achieving remarkable results.
Some of the most widely used SGD-based algorithms with an adaptive learning rate is AdaGrad [10]. In AdaGrad, the learning rate is adjusted dynamically on the basis of gradients computed in the previous iterations. The learning rate is updated as follows:

$$\begin{cases} g_t = \frac{\partial L(\theta_t)}{\partial \theta}, \\ V_t = \sqrt{\sum_{i=1}^{t} (g_i)^2 + \epsilon}, \\ \theta_{t+1} = \theta_t - \eta \frac{g_t}{V_t}, \end{cases} \tag{6.13}$$

where g_t is the gradient of the parameter θ at iteration t, V_t expresses the gradient history at iteration t, and θ_t is the value of the parameter θ at iteration t.
To sum up, compared to Gradient Descent, AdaGrad implements an automatic update of the learning rate based on the historical gradients accumulated.
AdaGrad presents two issues: first, the global learning rate η needs to be set manually and, second, the accumulated gradient increases indefinitely through the iterations and, thus, the learning rate will tend to zero and the update of the parameters will be ineffective.
Because of this, other algorithms were developed to improve AdaGrad performances. Some examples are AdaDelta, RMSProp and Adam [50]. In this work, we used Adam (Adaptive moment estimation) algorithm and for this reason we will introduce it more in detail.

Adam: Adam [28] is an advanced SGD method. Adam combines the adaptive learning rate method with another strategy called the momentum method. The latter method considers only the gradients that lie in a pre-defined window and uses the exponential moving average to calculate the exponentially decaying average of past gradients m_t and of squared gradients V_t:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{6.14}$$

$$V_t = \sqrt{\beta_2 V_{t-1} + (1 - \beta_2)(g_t)^2}, \tag{6.15}$$

where β_1 and β_2 are exponential decay rates. In addiction, in Adam, each parameter to be learned has its own adaptive learning rate. The formula to update the parameters θ is:

$$\theta_{t+1} = m_t - \eta \frac{\sqrt{1 - \beta_2}}{-\beta_1} \frac{m_t}{V_t + \epsilon}. \tag{6.16}$$

The default values of β_1, β_2 and ϵ are suggested to be set to 0.9, 0.999 and 10^{-8}, respectively. In practice, Adam is a common choice in the context of ML as it performs well and can be favourably compared to other adaptive learning rate algorithms [50].

6.5 Derivative-free Optimization

Derivative-free optimization is a discipline of mathematical optimization that allows to find the optimal solution even though the gradient of the objective function is not available (for instance in cases in which it is too much difficult to compute or it does not exist). In this field, there are two strategies that can be implemented. The first method follows empirical rules and, thus, it is based on the experience with the use of methods that have already been used in other contexts and that achieved significant results. Since the theoretical support for heuristic methods is usually weak, sometimes it is more desirable to adopt another method that consists in fitting a function according to the samples of the objective function, attaching some constraints to the considered search space in order to derive the samples.
To sum up, derivative-free algorithms do not require the detailed knowledge of the functional relationship between the parameters to be adjusted (optimized) and the loss function being minimized that is required in gradient-based algo-

rithms.
In the next Section, we will describe the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm, that we used in this work. Even though in the context of a band-limited AWGN channel, the derivative can be computed easily, we decided to implement this algorithm in the view of expanding this work to more complex (realistic) channels.

6.5.1 SPSA

Simultaneous Perturbation Stochastic Approximation (SPSA) [47], [48] is an optimization algorithm that belongs to the class of stochastic approximation derivative-free algorithms. This method involves an iterative procedure in which the parameters to be optimized are updated on the basis of the estimation of the gradient obtained from a random perturbation of the parameters of the objective function.
In order to understand how it produces the estimates of the gradient and how it updates the parameters, let us provide a summary of it.

- Step 1: *Initialization of the parameters.* Counter k is set to 1. a, c, A, α, and γ are initialized with non-negative coefficients, following the guidelines provided by the author [46]. Then they are used to compute the gain sequences $a_k = \frac{a}{(A+k)^\alpha}$ and $c_k = \frac{c}{k^\gamma}$; this choice is very critical to the performance of SPSA.

- Step 2: *Simultaneous perturbation vector.* A p-dimensional random perturbation vector Δ_k is generated from a zero-mean probability distribution. A straightforward (and theoretically-robust) choice is that of using a Bernoulli ± 1 distribution where $\pm$ 1 outcomes have equal probability ($\frac{1}{2}$).

- Step 3: *Evaluate the loss function.* At the k-th iteration, the current parameter vector is θ_k. Perturbing its value with the computed simultaneous perturbation vector Δ_k, we obtain two versions of the loss function $\mathcal{L}(\cdot)$: $y(\hat{\theta}_k + c_k\Delta_k)$ and $y(\hat{\theta}_k - c_k\Delta_k)$.

- Step 4: *Approximate the Gradient.* The approximation of the gradient is generated on the basis of the two obtained loss functions as follows:

$$\hat{g}_k(\theta_k) = \frac{y(\hat{\theta}_k + c_k\Delta_k) - y(\hat{\theta}_k - c_k\Delta_k)}{2c_k} \begin{bmatrix} \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ \vdots \\ \Delta_{kp}^{-1} \end{bmatrix} \tag{6.17}$$

  where Δ_{ki} is the i-th component of the Δ_k vector.

- Step 5: *Update θ estimate.* Then the estimation of the parameter θ at iteration $k+1$ is computed as follows:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k). \tag{6.18}$$

- Step 6: *Iteration or end condition.* Go back to Step 2) replacing k with $k+1$. If several successive iterations exhibit only a little change or the maximum allowable number of iterations has been reached, the algorithm terminates.

Relying on these guidelines, we implemented it in our framework.
Even though it is difficult to compare gradient-based with gradient-free methods due to the different information needed, as a general rule, the gradient-based algorithms will reach the convergence faster than the gradient approximations (assuming to measure the speed with the number of iterations). In particular, it is possible to derive the optimum rate of convergence that depends on how much the obtained parameter estimate is far from the true optimal parameter θ. Given k the number of iterations, it is proportional to $k^{-\frac{1}{2}}$ for the gradient-based algorithms and to $k^{-\frac{1}{3}}$ for gradient-free algorithms, thus, as we said, the former is faster than the latter because of the additional information that is used in gradient-based algorithms [48].

7

System basic structure and conventional equalization method: design and analysis

7.1 Outlook on the work

The simulation software developed for this study is written in Python. The contents include Jupyter notebooks and code modules. For the ML-based part, we used the Pytorch framework, SciPy and other Python libraries.
The project objectives are 1) the design and 2) the validation of a non-conventional ML-based technique aimed at compensating for the distortion of a transmitted signal caused by a communication channel with impairments.
To achieve these goals, in this work, we will implement two different setups:

1. the first is the actual ML-based structure whose purpose is to optimize the PS and/or MF to compensate for the channel impairments without implementing conventional digital signal processing blocks/algorithms;

2. the second is a more conventional setup with a post-equalizer that will be used as a benchmark for the ML-based solution.

The ML framework will be presented in detail in the next Chapter 8. While here, in Chapter 7, we will present the conventional setup that we will use to validate the proposed ML-based framework. More specifically, in this Chapter, first, we will describe the design of the communication system with the AWGN

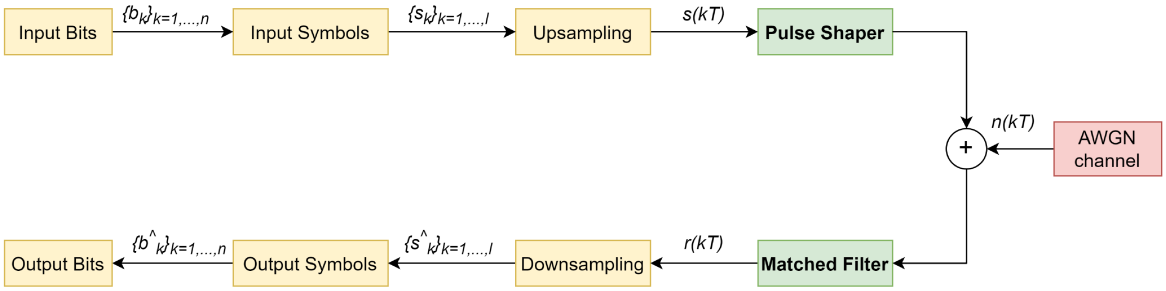


Figure 7.1: Digital communication system setup with Pulse Shaper, AWGN channel and Matched Filter.

channel, then we will add a Low-Pass Filter (LPF) as a source of impairment and, finally, we will present the conventional post-equalization strategies we used.

Notation

Assuming a discrete-time baseband equivalent model, the first step was the implementation of a traditional AWGN channel simulator.
In this section, we will present the notation we will use to describe the blocks of the communication system with the AWGN channel that we can see in Figure 7.1.
Within this Chapter, we will refer to:

- n: the number of bits that we want to transmit;
- $\{b_1, b_2, ..., b_n\}$: the binary sequence generated randomly;
- M: the cardinality of the modulation format;
- $l = \frac{n}{\log_2 M}$: the number of transmitted symbols.
- $\{s_1, s_2, ..., s_l\}$: the symbols to be transmitted;
- sps: the samples per symbol;
- T_s: the symbol period;
- R_s: the symbol rate;
- $f_{sampling}$: the sampling frequency;
- $T = \frac{T_s}{sps}$: the sample period;
- $s_i(kT)$ with $i \in \{I, Q\}$: samples of the In-Phase or the Quadrature signal after the upsampling and the convolution with the Pulse Shaper;
- $n_i(kT)$ with $i \in \{I, Q\}$: I or Q AWGN samples;

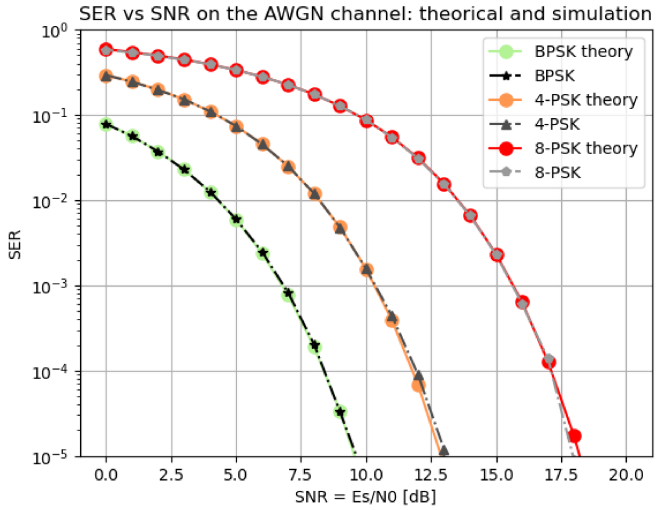


Figure 7.2: SER vs SNR: theoretical and Montecarlo simulation curves.

- $s_i(kT) + n_i(kT)$ with $i \in \{I, Q\}$: the sum of the transmitted signal and the AWGN noise relative to I or Q channel;
- $r_i(kT)$ with $i \in \{I, Q\}$: the received signal after the convolution with the Matched Filter in the I or in the Q channel;
- $\{\hat{s}_1, \hat{s}_2, ..., \hat{s}_l\}$: the detected symbols;
- $\{\hat{b}_1, \hat{b}_2, ..., \hat{b}_n\}$: the binary sequence corresponding to the detected symbols;
- L_{RRC}: the length of the conventional RRC and of the FIR that we want to learn;
- $\{h_k^{RRC}\}_{k=1,...,L_{RRC}}$: the taps of the conventional RRC filter;
- $\{w_k\}_{k=1,...,L_{RRC}}$: the parameters of the FIR filter that we want to learn during the training.

For the simplicity of the notation, from now on we will refer only to the In-Phase components when we will deal with IQ samples.
The first version of the simulator includes a pulse shaper, a block that implements the additive white Gaussian noise and a matched filter. The binary sequence was generated randomly and the adopted modulations were BPSK, QPSK and 8-PSK.
In Figure 7.1, we can see the block diagram: first we have the generation of bits $\{b_k\}_{k=1,...,n}$, then they are mapped into symbols $\{s_k\}_{k=1,...,l}$; afterwards there is the pulse shaper that returns the signal $s(kT)$ that is injected into the AWGN

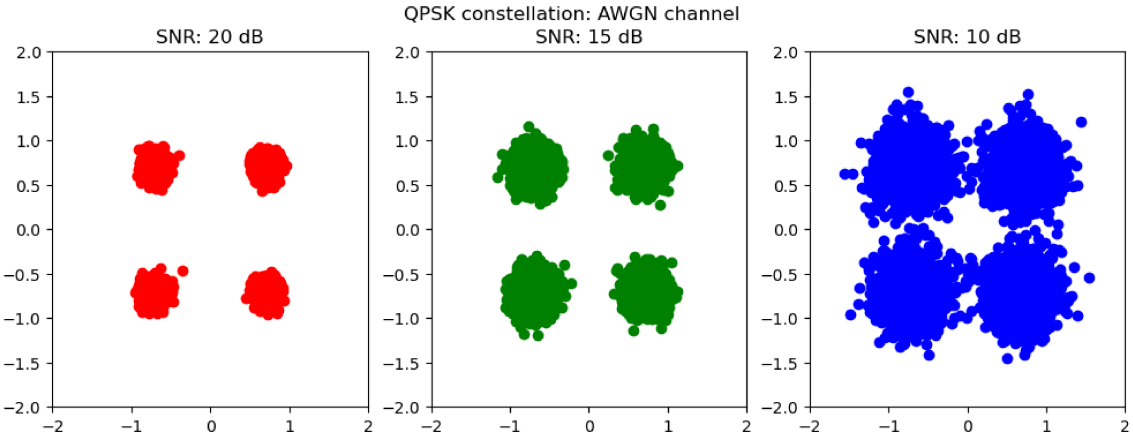


Figure 7.3: QPSK constellation at different SNR values (10 dB, 15 dB and 20 dB).

channel. After the propagation along the AWGN channel, the noisy signal represented as $s(kT) + n(kT)$ is convolved with the matched filter and the resulting signal $r(kT)$ is downsampled. Then, the detection is performed using the received IQ samples and the detected symbols $\{\hat{s}_k\}_{k=1,\dots,l}$ are obtained. Afterwards, the Symbol Error Rate (SER) is computed and it is compared with the theoretical SER, established by the Shannon theorem in an AWGN channel. Figure 7.2 shows the theoretical curves for the SER at a certain SNR compared to the results of Montecarlo simulations performed on 100000 symbols and averaged on a certain number of iterations. The two curves are compatible between each other within the errors of a numerical simulation run on a dataset of 10^5 symbols; this match can be considered as a validation of our simulator and, thus, we have a robust system on which we can add new blocks and features.

7.2 AWGN CHANNEL SETUP

In this section, we will briefly describe the design of the blocks of the communication system that we can see in Figure 7.1. We will focus on the PS/MF blocks, describing their functionalities and their importance in communication systems.

SYMBOL GENERATION

We implemented the M-ary Phase Shift Keying (MPSK) modulation method where M is the cardinality of the selected constellation. In particular, we im-

plemented BPSK, QPSK and 8-PSK modulation formats with 2, 4 and 8 symbols respectively. We will refer to the constellation symbols with the following notation: $\{s_k^{MPSK}\}_{k=1,\dots,M}$.
In Figure 7.3, we show the constellation plot of the QPSK affected by different values of AWGN. With more complex simulation setup, from now on, we will show only QPSK simulation and for results connected to IQ samples (for instance, the discrete-time waveforms or the Eye-diagrams) we will show the In-Phase samples only for simplicity, as the Quadrature samples lead to comparable results.

Upsampling

Upsampling is used in pulse shaping with the aim of creating smoother waveforms for the transmission through the medium.
In order to map the symbols $\{s_k\} \in \mathbb{C}$ to their upsampled version, we took the I and the Q components individually and we appended $sps-1$ samples initialized to zero to each of them before the pulse shaper at the transmitter.

Pulse Shaper

In communication systems, the pulse shape should provide the two following functionalities:

- generating band-limited channels;
- reducing ISI.

Indeed, the pulse shaping allows to minimize sharp transitions of the signal, bounding the frequency components within a certain range, with the aim of achieving spectral and power efficiency [14]. In addiction, the pulse should have a specific shape such that it should exhibit zero crossing at all the sampling points except one, i.e. the instant that corresponds to the symbol associated to the pulse [30]. This condition is formalized by the Nyquist criterion, that prevents from ISI, avoiding the overlapping of adjacent symbols at the optimum sampling point.

Matched Filtering paradigm It is convenient to split the pulse shaping operation equally between the transmitter and the receiver since both can take advantage of the low-pass filter property of the pulse shaping: at the transmitter it is

desirable to have a low-pass filter to reduce the amount of spectrum used by the signal and at the receiver the low-pass filter is advantageous to remove from the signal as much noise/interference as possible. This strategy is called "matched filtering".
In our setup, we used a Root-Raised Cosine (RC) pulse shape that is the most typically used since it provides a good tradeoff between spectral efficiency and design complexity [24], [59]. We also adopted the matched filtering paradigm, splitting the RC into two identical RRC pulse shapes.
In the next section, we will provide the function of the RC, from which we will derive the RRC formula.

The RC and the RRC Pulse Shapes

RC filter Given T_s, the symbol period, and β, the roll-off factor, in the frequency domain, the RC filter is described by the following formula:

$$H_{RC}(f) = \begin{cases} 1, & |f| \leq \frac{1-\beta}{2T_s} \\ \frac{1}{2}\left[1 + \cos\left(\frac{\pi T_s}{\beta}\left[|f| - \frac{1-\beta}{2T_s}\right]\right)\right], & \frac{1-\beta}{2T_s} < |f| \leq \frac{1+\beta}{2T_s} \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

The RC plays a fundamental role in digital communication system because of its ISI-free property (known as Nyquist theorem), that can be expressed as follows:

$$h_{RC}(nT_s) = \delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n = \pm 1, \pm 2, \ldots \end{cases} \tag{7.2}$$

where $H_{RC}(f)$ is the frequency response and $h_{RC}(t)$ is the time response of the channel.
As we said in the previous section, in practical communication systems, the RC filter is split between the transmitter and the receiver.

Splitting a Filter in Half The combination of the PS and of the MF is a convolution. Recalling that the convolution corresponds to a multiplication in the frequency domain:

$$h_{RRC}(t) * h_{RRC}(t) \longleftrightarrow H_{RRC}(f) \cdot H_{RRC}(f), \tag{7.3}$$

in order to split the RC filter between the transmitter and the receiver side, it is possible to take the square root of the frequency response. Therefore, the frequency response $H_{RRC}(f)$ of the transmitted waveform $h_{RRC}(t)$ should be the square root of $H_{RC}(f)$, i.e.,

$$\begin{aligned} H_{RRC}(f) &= \sqrt{(H_{RC}(f))} \\ h_{RRC}(t) &= \mathcal{F}^{-1}(H_{RRC}(f)) \end{aligned} \tag{7.4}$$

where $\mathcal{F}$ is the Fourier transform and $\mathcal{F}^{-1}$ its inverse.
Splitting the RC filter between the transmitter and the receiver allows to take advantage of the band-limiting capability of the RRC. Moreover, because of the associative property of the convolution, their combined effect is the same as a single RC filter and, thus, the ISI-free condition is met.

RRC filter Computing the inverse Fourier Transform of Equation 7.4, we obtain the impulse response of a RRC:

$$h_{RRC}(t) = \begin{cases} \frac{1}{T_s}\left(1+\beta(\frac{4}{\pi}-1)\right), & t=0 \\ \frac{\beta}{T_s\sqrt{2}}\left[\left(1+\frac{2}{\pi}\right)\sin\left(\frac{\pi}{4\beta}\right)+\left(1-\frac{2}{\pi}\right)\cos\left(\frac{\pi}{4\beta}\right)\right], & t=\pm\frac{T_s}{4\beta} \\ \frac{1}{T_s}\dfrac{\sin\left[\pi\frac{t}{T_s}(1-\beta)\right]+4\beta\frac{t}{T_s}\cos\left[\pi\frac{t}{T_s}(1+\beta)\right]}{\pi\frac{t}{T_s}\left[1-\left(4\beta\frac{t}{T_s}\right)^2\right]}, & \text{otherwise} \end{cases} \tag{7.5}$$

The pulse shape that is used in real setup is the discrete-time version of the RRC pulse shape [58]. As an example, in Figure 7.4, we can see a RRC with roll-off $\beta = 0.6$ and $T_s = 0.25$ s that we selected for several simulations.

PS ENERGY NORMALIZATION

In our AWGN channel setup, the RRC was normalized in order to have unitary energy $E_{h_{RRC}}$. As a result, the energy of the raw symbol and its upsampled version after the PS will be the same.

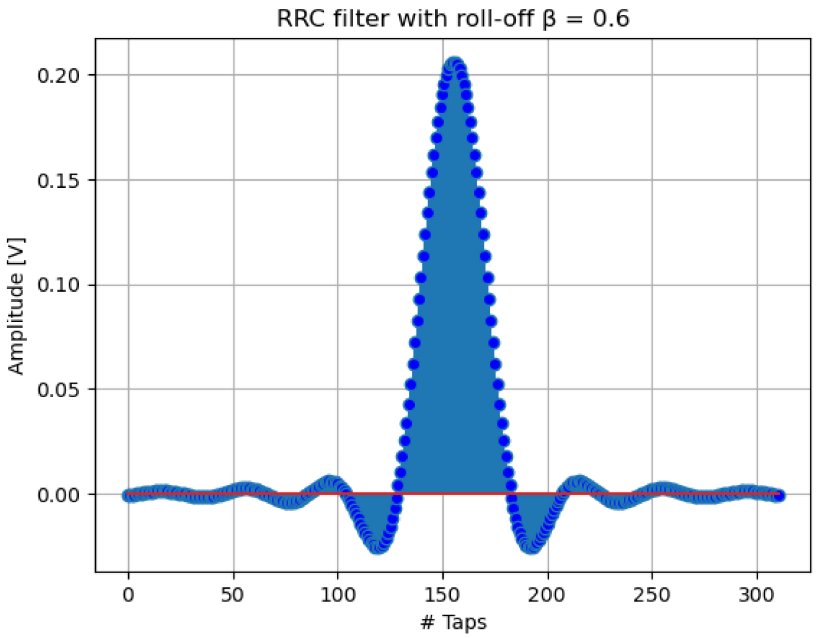


Figure 7.4: Root-raised cosine with $\beta = 0.6$ and $n = 311$ taps in time-domain.

The taps of the RRC were normalized so that:

$$E_{h_{RRC}} = \sum_{l=1}^{L_{RRC}} |h_l^{RRC}|^2 = 1 \tag{7.6}$$

where L_{RRC} is the length of the RRC filter.
Thanks to this normalization, when we compute the SNR, we will always assume to start from a scenario where the symbol energy E_s is equal to 1.

AWGN CHANNEL

After the pulse shape block, the signal propagates along a channel and is affected by the AWGN.
AWGN is described by its PSD N_0 expressed in $\frac{V^2}{Hz}$. It is modelled as a Gaussian random variable $\mathcal{N}(\mu, \sigma)$ with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{\frac{N_0}{2}}$ for both I and Q samples.
With the aim of computing the SNR at which we operate in the communication system, we would need to compute the energy of the noise, multiplying the PSD N_0 by the bandwidth B occupied by the signal. However, in order to generalize our results, we will use a normalized SNR measure, the so-called energy

per symbol to noise power spectral density ratio. In order to be consistent with all the modulation formats, we defined the energy per symbol to noise power spectral density ratio as the ratio between the energy of a symbol and the power spectral density of the noise, considering both I and Q channels:

$$\frac{E_s}{N_0} = \frac{\text{symbol energy}}{\text{noise power spectral density}}. \tag{7.7}$$

Given that the symbol energy is E_s is fixed to 1, the PSD of the noise N_0 can vary to obtain a specific value of the SNR in a range that spans from 0 dB up to 40 dB. The selected SNR range provides an insight into SER values that are much smaller than the precision that we are able to provide in our simulations that is around 10^{-5}. This value was dictated by the order of magnitude of the samples we used within the simulations. Indeed, because of the limited computational resources, we fed our system with samples of 10^5 symbols at maximum. Therefore, assuming to use the QPSK modulation format, the SNR useful range for the computation of the SER goes from 0 dB to 12 dB because when the SNR is at 12 dB, the SER value is in the order of 10^{-4} on the AWGN channel (as we can see in Figure 7.2). However, even though we were not able to compute an accurate SER beyond this SNR value, we were still able to evaluate other metrics, such as the MSE loss and the Eye-diagrams and, thus, we decided to train and to test our model up to 40 dB, to get insight into higher SNR values where the impact of noise is much lower.

Match Filter and Downsampling

Going on with the scheme we have in Figure 7.1, after that the AWGN block was added to the signal, we convolved the noisy signal with the MF that is a RRC matched with the PS. Afterwards, we downsampled the signal, selecting only one sample per symbol.
Downsampling is an operation used in signal processing in which the sampling rate of a discrete-time signal is reduced by removing samples. In this case, we downsampled the received signal $r(kT)_{k\in\mathcal{Z}}$ of a factor of $sps = 32$, and, thus, we only maintained the samples $r(m \cdot sps \cdot T)_{m\in\mathcal{Z}}$.

Property	Symbol	Value
Symbol-rate	R_s	4 Hz
Symbol time	T_s	$\frac{1}{R_s} = 0.25$ s
Roll-off factor	β	0.6
Sampling frequency	$f_{sampling}$	128 Hz
Samples per symbol	sps	32
RRC length	L_{RRC}	311
Spectral width of RRC	B_{RRC}	$\frac{1}{2}\frac{1+\beta}{T_s} = 3.2$ Hz
LPF order	$order$	2
LPF critical frequency	$f_{critical}$	$[0.5, 0.295, 0.273] \cdot B_{RRC}$
Energy per symbol to noise PSD	$\frac{E_s}{N_0}$	$[0, 1, ..., 19, 20]$ dB

Table 7.1: Simulation parameters.

Symbol detection

In order to detect the transmitted symbol, we applied the Maximum Likelihood criterion that in the context of AWGN is equivalent to the Minimum Distance criterion [3].
Therefore, given the constellation symbols $\{s_k^{MPSK}\}_{k=1,...,M}$, we computed the Euclidean distance between each symbol of the constellation and the received IQ samples, obtained from the two downsampled signals $r(m \cdot sps \cdot T)_{m \in Z}$, one relative to the I and one to the Q channel. The symbol with the minimum distance will be selected as the detected symbol $\{\hat{s}_m\}_{m=1,..,l}$.

7.3 Simulation parameters

In the developed framework, there are many parameters that we need to set in order to run the simulation. Some of them were chosen quite arbitrarily since it was sufficient that the ratio between them was matched. In Table 7.1, we present a specific choice of parameters that we used to run the simulations whose outcomes are displayed in the Figures of this Chapter. The first parameters we set were the symbol rate R_s and the roll-off factor β of the traditional RRC that we have at the PS and MF of the AWGN channel simulator. The roll-off factor β is a measure of the excess bandwidth occupied by the transmitted

waveform, whose spectral width can be computed as follows:

$$B_{RRC} = \frac{1}{2}\frac{1+\beta}{T_s} = 3.2Hz. \tag{7.8}$$

Afterwards, we set the sampling frequency parameter $f_{sampling}$, that is the rate at which the symbol is sampled at the receiver. In the design of a real communication system, an essential condition that has to be matched is the Nyquist sampling theorem that states that the sampling frequency $f_{sampling}$ must be greater or equal to twice the bandwidth B_{RRC} occupied by the considered signal. Applied to this specific scenario, we can express it as follows:

$$f_{sampling} \geq 2B_{RRC}. \tag{7.9}$$

In order to operate in a regime in which the Nyquist sampling theorem is met, we set $f_{sampling}$ to 128 Hz. As a result, we will have $\frac{f_{sampling}}{R_s} = 32$ samples per symbol and, thus, the Nyquist sampling theorem will be matched: 128 Hz ≥ 6.4 Hz. Another important aspect to evaluate in the design is the condition for zero-ISI. In the traditional AWGN channel scenario, this condition, i.e. the Nyquist criterion, is set by design because we selected the RC pulse shape - split as a RRC at transmitter and a matched RRC at the receiver - that belongs to the family of the Nyquist pulse shapes and has been designed exactly to achieve this purpose [59]. As we discussed in the previous section, RC meets indeed the zero-ISI condition as it exhibits zero crossing at all the sampling points except at the optimum sampling point of the considered symbol [30].
Let us disclose that, in the design of the ML-based framework, the condition for zero-ISI will be a fundamental condition to satisfy. Indeed, in order to achieve the purpose of optimizing the PS and/or the MF, the convolution between the two filters and the LPF impulse response has to be a Nyquist pulse shape able to avoid the overlapping of adjacent symbols at the optimum sampling point, preventing from intersymbol interference.
Going back to the AWGN channel setup, since both the sampling theorem and the Nyquist condition are met in our design, at the receiver we will prevent aliasing and also maintain the integrity of the signal.
As we discussed in Chapter 4, currently, there is a limited knowledge about optimal data representation, loss function, and training strategies, etc., in the context of ML applied to Digital Communications. In absence of previous expertise, we

spent a lot of time to find the proper length of filters to represent the data. On the one hand, the number of taps should be sufficient to represent the signals. For instance, the RRC is an infinite time signal, but since we are representing it in a finite system, we accurately tuned its length to show enough ripples to clearly see the effect of ISI in the band-limited channel. We followed the recommendations we found in [58] that states that - for realizability considerations - the discrete-time transmit pulse shaping filter and receiver matched filter with a modulation with a small cardinality (as in the considered case) should have at least a length that spans $\pm 4T_s$ in order to perform a proper simulation. In the considered case, $\pm 4T_s$ corresponds to $\pm 4 \cdot 32 = \pm 128$, that is 256 samples in total. For this reason, we started to explore some possible values for the length of our filter starting from 256.
On the other hand, at the same time we didn't want to employ a too large filter, first, because the adaptive FIR filter would have had unnecessary taps to learn (and we do not want to overcomplicate the problem) and this would lead to undesired results for the simulation and also not desirable in practice, since a longer FIR length implies a higher hardware implementation cost.
Many choices and combinations were explored, but for the purpose of doing a complete analysis, we selected a specific combination of parameters using the *GridSearch()* algorithm. As an example, in Figure 7.5 we plotted the SER computed at SNR = 12 dB for different lengths of the trainable FIR filter placed at the transmitter. As we can see, the length of the trainable filter influences the SER. At $L_{RRC} = 311$ we achieved a slightly lower SER than the other considered values and for this reason we decided to use it in our simulation. Similar considerations were done for other parameters like the hyperparameters of the ML-framework.

7.4 Band-limited AWGN channel setup

In order to emulate a bandwidth-limited receiver and to move to a more realistic - but still easy to understand - channel, we added a LPF to the basic system with AWGN. Figure 7.6 shows the block diagram we used in this new setup.
In general, in order to characterize a LPF, we need to specify the cutoff frequency f_{cutoff}, that is the frequency at which the filter gain is $\frac{1}{\sqrt{2}}$ (that corresponds to -3 dB) with respect to the passband (unitary) gain (i.e. 0 dB). Depending on the value of f_{cutoff} with respect to the bandwidth of the transmitted signal (in our

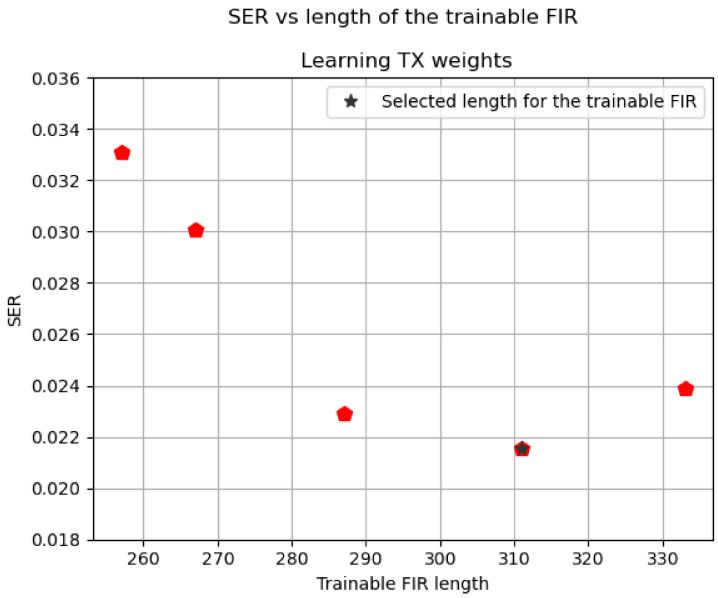


Figure 7.5: Example usage of *GridSearch()*: SER VS length of the trainable FIR.

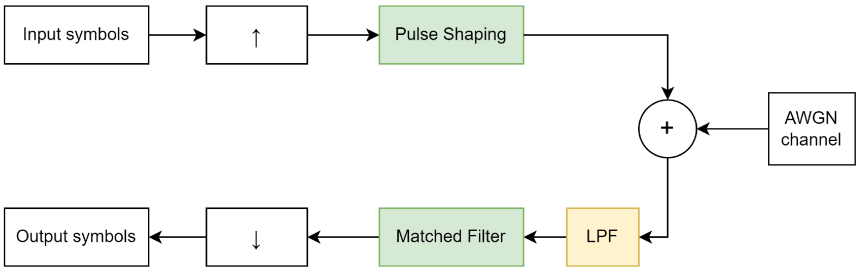
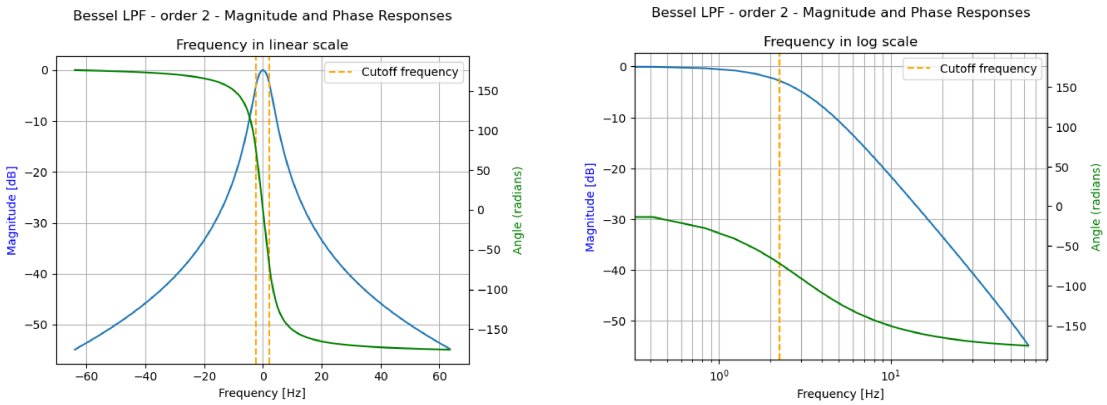


Figure 7.6: Band-limited AWGN channel simulator.



(a) Frequency in linear scale.

(b) Frequency in log scale.

Figure 7.7: Bessel LPF - order 2 - Magnitude and Frequency responses.

case B_{RRC}), the low-pass filter can cause the distortion of the transmitted signal. Given $C(f) = |C(f)|e^{j\theta(f)}$ the channel frequency response, there are two types of channel distortion that may affect the transmitted signal:

- an amplitude distortion, when the amplitude characteristic $|C(f)|$ is not constant for $|f| < B_{RRC}$;
- a phase distortion, that occurs when the phase characteristic $\theta_c(f)$ is non-linear in frequency for $|f| < B_{RRC}$. Equivalently, the phase distortion can be expressed by the envelope delay characteristic - defined as $\tau(f) = -\frac{1}{2\pi}\frac{d\theta_c(f)}{df}$. If $\theta_c(f)$ is linear in f, then the envelope delay $\tau(f)$ is constant for $|f| < B_{RRC}$.

In the next Section, we will analyze the specific LPF we selected for our setup.

7.4.1 Bessel LPF

Starting from the implementation of the AWGN channel setup, we added an analog low-pass Bessel filter. We selected it because it enables signal transmission with very low signal deformation which makes it a good match for high speed digital transmissions systems [52]. The Bessel filter is indeed optimized to provide constant group delay $\tau(t)$ in the filter passband. As a result, it has a low dispersion and, thus, it allows filtering with minimal linear distortions [53]. In Figure 7.7, we can observe the Bessel filter frequency response divided into magnitude and phase responses.
The presence of the LPF with a sufficiently small cutoff frequency f_{cutoff} causes the distortion of the signal, i.e. the occurrence of intersymbol interference (ISI)

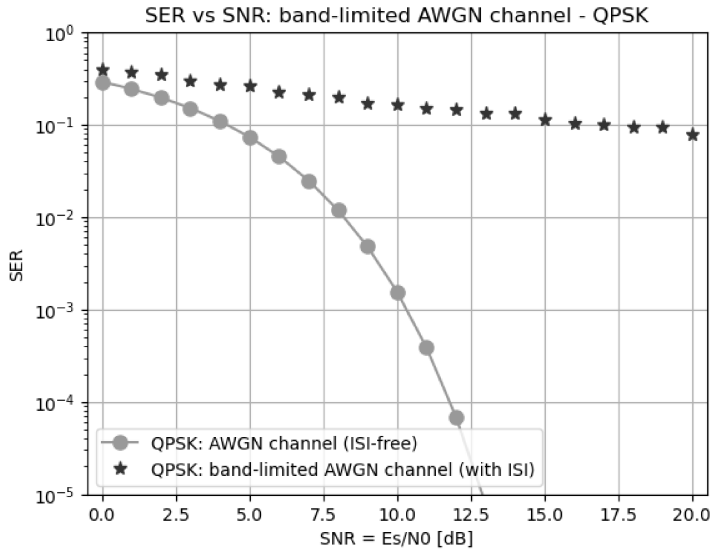
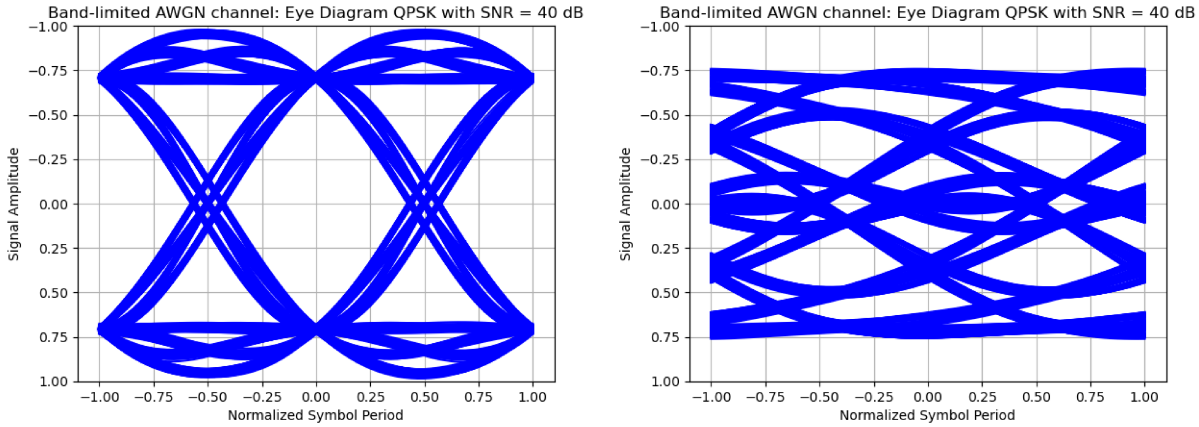


Figure 7.8: SER vs SNR: band-limited AWGN channel with QPSK constellation.

that results into a penalty in terms of Symbol-Error-Rate (SER) as we can see in Figure 7.8. The presence of ISI with the overlap of adjacent symbols can be clearly noticed in Figure 7.9 that shows the Eye-diagram displayed at 40 dB in the ISI-free AWGN channel (on the left) that we can compare with the Eye-diagram in the band-limited AWGN channel where ISI occurs (on the right).

Bessel filter design

To implement the Bessel filter, we used the *scipy.signal* library for DSP. In order to define it, the *scipy.signal* function we used required to set parameters such as the filter's order - that defines the steepness of the transfer function - and its critical frequency $f_{critical}$. The Bessel filter critical frequency $f_{critical}$ is defined with respect to the cutoff frequency f_{cutoff} of a Butterworth filter: it corresponds to the f_{cutoff} of a Butterworth filter of the same order that shares the same response asymptotes. In Figure 7.11, we can see the frequency response of a Bessel filter whose critical frequency corresponds to the cutoff frequency of a Butterworth filter of the same order.
In the next paragraph, we will analyze the interplay of the AWGN and the limited bandwidth of the LPF in order to make a proper choice of the critical frequency $f_{critical}$ to balance the amount of bandwidth that the LPF is cutting.



(a) Eye-diagram AWGN channel: ISI-free.

(b) Eye-diagram band-limited AWGN channel: with ISI.

Figure 7.9: Eye-diagrams (In-Phase component): comparison between AWGN channel (ISI-free) and band-limited AWGN channel (with ISI).

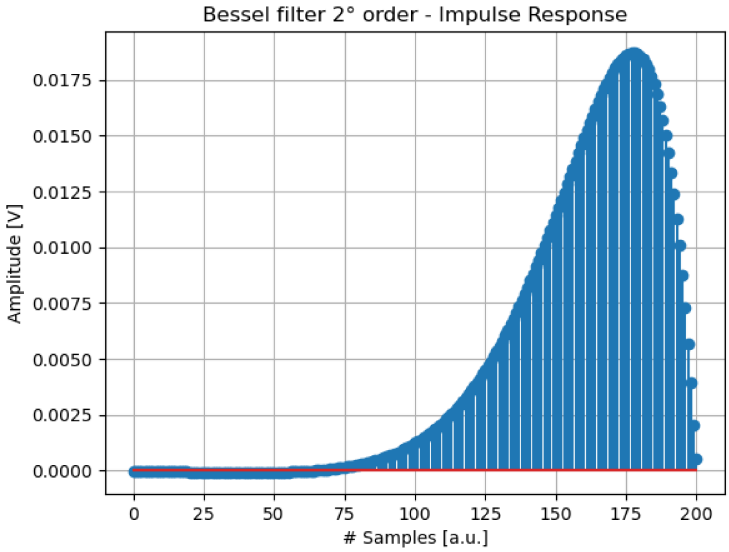


Figure 7.10: Impulse Response of Bessel LPF - order 2.

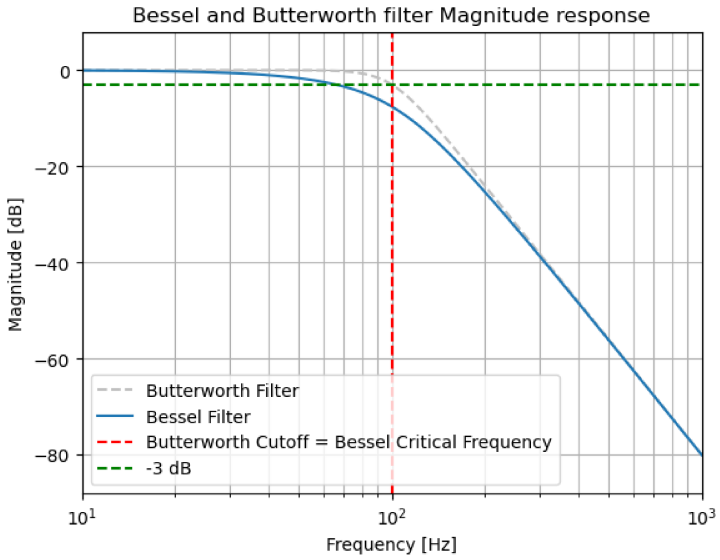


Figure 7.11: Comparison between Bessel and Butterworth Magnitude Response.

The choice of the critical frequencies

In order to select the critical frequencies $f_{critical}$ for our analysis, we started with several values and we computed the SER in presence of both LPF and AWGN channel (band-limited AWGN channel) with a SNR equal to 10 dB and in presence of the LPF only (noiseless band-limited channel) in order to show the penalty due to the stand-alone Bessel filter.
In Figure 7.12, on the x axis we have the critical frequencies and in the y axis we represent the SER in the band-limited AWGN channel and in the noiseless band-limited channel. It can be noticed that, for low values of the critical frequency, the two SER coincide; it means that the Bessel filter is cutting a significant part of the spectrum of the transmitted signal and, thus, the LPF is completely responsible for the penalty that is present in the channel, predominating on the AWGN.
As soon as the critical frequency increases, we can see that the two SER have different values and this means that we can clearly distinguish between the two contributions that affect the channel and cause the SER. We selected the critical frequency $f_{critical} = 111.82$ Hz as representative of this behaviour (yellow dashed line in the Figure). Increasing even more the critical frequency, we can see that the penalty due to the LPF itself decreases and then disappears from the Figure (see the blue dashed line), going below a SER of 10^{-5}, that is the smallest value that we plotted. However, even though the stand-alone LPF does not

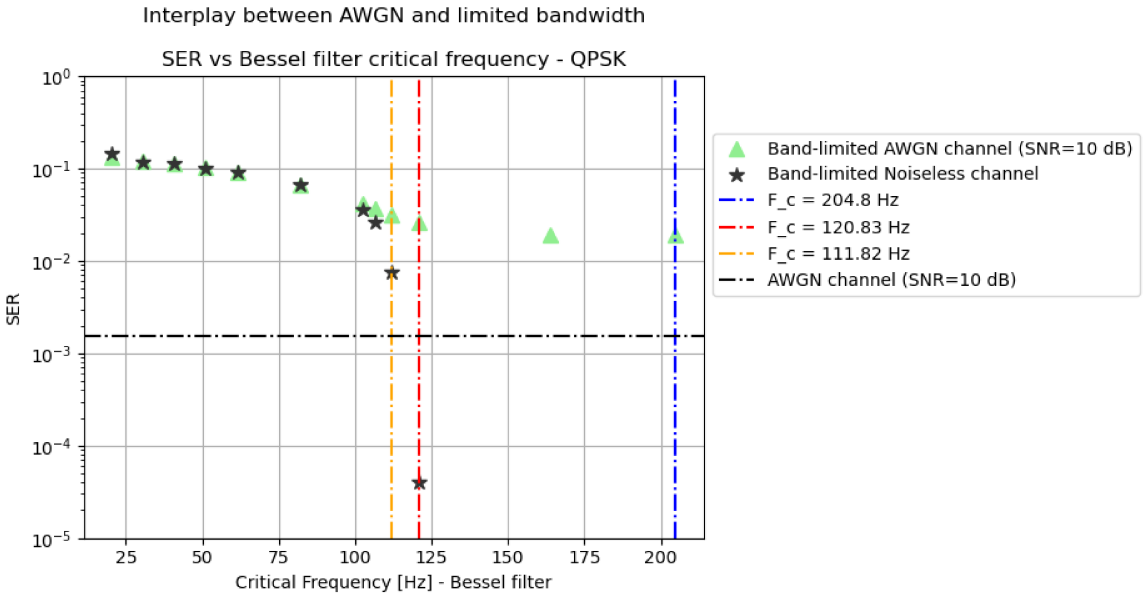


Figure 7.12: SER VS Bessel filter Critical frequency

Critical Frequency
$0.5 \cdot B_{RRC} = 204.80$ Hz
$0.295 \cdot B_{RRC} = 120.83$ Hz
$0.273 \cdot B_{RRC} = 111.82$ Hz

Table 7.2: Critical frequencies.

introduce errors within a SER of 10^{-5}, it is still present in the channel and the combination between its effect and the AWGN shows a penalty with respect to the AWGN channel curve. For this reason, for the simulations, we have chosen a value of the critical frequency that implies a low penalty (blue dashed line) and two values with a slightly higher penalty (red and yellow dashed lines). The frequencies will be expressed as a fraction of the bandwidth occupied by the RRC B_{RRC} as we can see in Table 7.2.
In Figure 7.13, we can see how the change of the Bessel filter critical frequency affects the SER. The first selected critical frequency $f_{critical} = 204.80$ Hz corresponds to a case in which we are cutting only a small portion of the spectrum and this results into a small channel impairment. For the other two critical frequencies $f_{critical} = 120.83$ Hz and $f_{critical} = 111.82$ Hz, we can notice that the presence of the Bessel filter has a much higher impact on the SER.

Basing on the results of this analysis, we selected the Bessel filter critical frequency $f_{critical} = 120.83$ Hz $= 0.295 \cdot B_{RRC}$ and we adopted it for the simulations

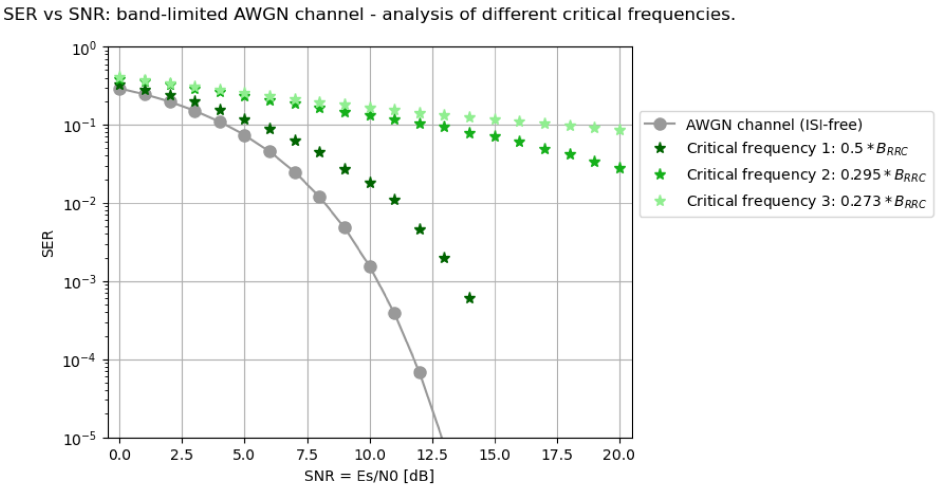


Figure 7.13: Band-limited AWGN channel: Bessel filter critical frequencies analysis.

relative to the results that we will present in this Chapter 7 and in the next Chapter 8.
In Figure 7.14, we can see the magnitude responses (on the left) and the impulse responses (on the right) of the raw RRC and of the convolution between the RRC and the LPF (with $f_{critical} = 0.295 \cdot B_{RRC}$). Observing the frequency domain plot (on the left of Figure 7.14), we can see that the LPF is cutting a significant portion of the RRC spectrum; this results into a distortion of the RRC shape in terms of phase and amplitude as we can notice from the time domain plot (on the right of Figure 7.14).

7.5 Post-equalization as a benchmark

As soon as we selected the proper simulation parameters, we implemented a conventional communication system with post-equalizers that will allow us to evaluate the results we got in the ML-based framework. In this way, we will be able to compare the two systems, the traditional and the ML-based.

7.5.1 Equalizer design

Two different implementations of the equalizer will be provided: the first will be realized at sample-level, while the second will be implemented at symbol-level.

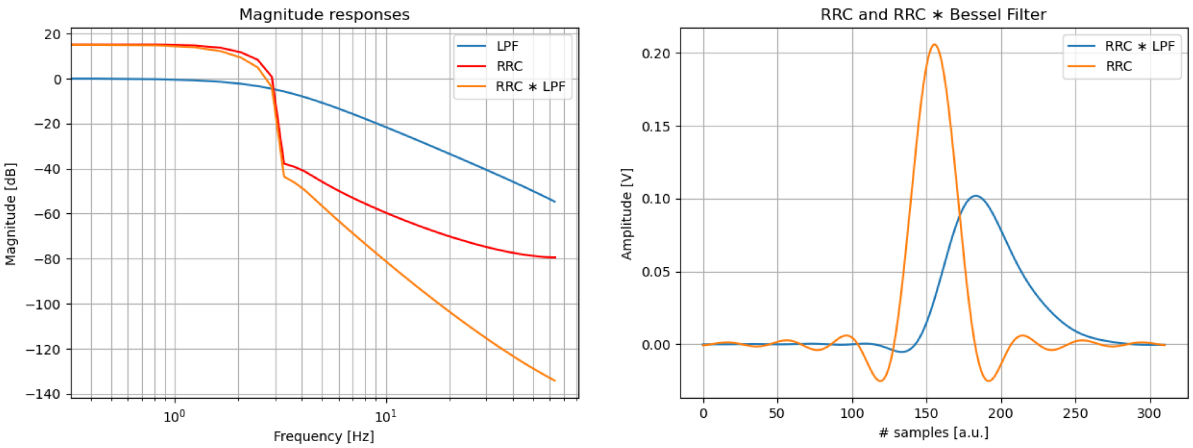


Figure 7.14: Magnitude Responses and Impulse Responses of RRC and RRC ∗ LPF.

Both the implementations use the LMS algorithm [18], [20], that we will provide below:

n: length of the input vector.

p: number of the filter taps.

$$\begin{aligned} x(n) &= [x(n), x(n-1), ..., x(n-p+1)]^T \\ h(n) &= [h_0(n), h_1(n), ..., h_{p-1}(n)]^T \\ y(n) &= h^H(n) \cdot x(n) \\ d(n) &= y(n) + v(n) \\ e(n) &= d(n) - \hat{y}(n) = d(n) - \hat{h}(n) \cdot x(n) \end{aligned} \tag{7.10}$$

For both cases, we implemented the process with the following procedure:

1. firstly, we implemented a system without AWGN provided with a linear post-equalizer based on the LMS algorithm that adjusted iteratively the parameters to invert the channel characteristic;

2. afterwards, we tested the obtained filter in a noiseless channel to check that the learned parameters were able to eliminate ISI;

3. then, we restored the band-limited AWGN channel for values of SNR that span from 0 dB to 20 dB and, for each SNR value individually, we performed the training again in order to optimize the filters in presence of noise;

4. eventually, we tested the obtained filters for each SNR value.

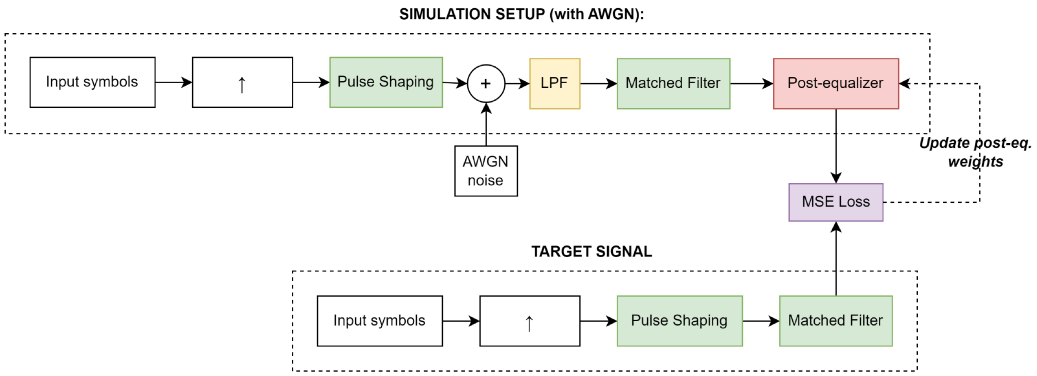


Figure 7.15: Block diagram of a post-equalizer at sample-level.

Sample-level

The equalization at sample-level is performed directly on the waveform samples and it is implemented as a Finite Impulse Response (FIR) filter. At the beginning, as we said, we learned the equalizer taps in absence of noise. As a result, we obtained the inverse of the LPF response. This implementation forces the ISI to be zero at the sampling time and for this reason this equalizer is called a zero-forcing equalizer. The drawback is that it is noise sensitive, so the lower the SNR, the higher the noise enhancement will be. It is exactly the effect that we can notice in Figure 7.16 where we can see the magnitude response of several post-equalizers sample-level that we obtained at different SNR values.
Describing Figure 7.16 more in detail, we can notice that the magnitude responses of the post-equalizer at different values of the SNR coincide between each other at high frequencies, while at low frequencies the amplification increases with the decrease of the SNR. In other words, the lower the SNR, the higher the PSD of the noise N_0 and, thus, the noise-sensitive post-equalizer will enhance not only the frequencies of interest to invert the channel, but also the noise as well.
For this reason, for each value of the SNR, we re-trained the post-equalizer in order to optimize the equalizer's parameters in presence of noise.
As a result, we were able to perform the equalization for the penalty introduced by the LPF almost completely. In Figure 7.17, we can see indeed that in presence of the sample-level post-equalizer the obtained SER for SNR values from 0 dB to 20 dB almost approaches the theoretical AWGN channel SER curve.

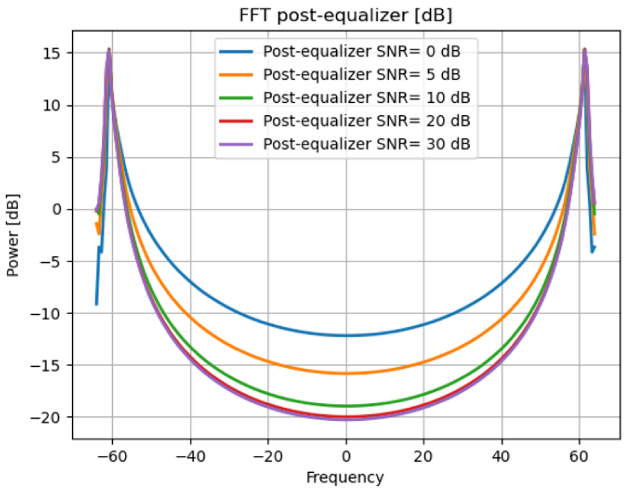


Figure 7.16: Post-equalizer at sample level obtained for different values of SNR.

This is due to the fact that we are operating at the finest resolution and, thus, the equalization is very precise and allows to compensate almost completely for the ISI that we experience at the receiver side.
The main drawback of a sample-level post-equalizer is the computational complexity that it implies and, thus, it is not usually adopted for real implementations.
For this reason, we decided to move to a symbol-level post-equalizer.

Symbol-level

In the view of extending this work to more complicated and realistic channels, we have to consider that many communication systems work with 2 or even fewer samples-per-symbol as we saw for the short-reach optical link in Chapter 3, where we presented the symbol-level FFE equalizer as one of the most commonly used equalization techniques.
Therefore, we decided to implement a post-equalizer at symbol-level. In Figure 7.18, we can see the symbol-level post-equalizer setup.
The main advantage of a symbol-level post-equalizer is that it reduces the computational complexity since the equalization is performed after the downsampling, so only one sample per symbol is considered.
In Figure 7.19, we can see the SER that we obtained in presence of the post-equalizer at symbol-level (Figure 7.19). We can notice that there is only a weak improvement in terms of SER w.r.t. to the penalty curve of the band-limited

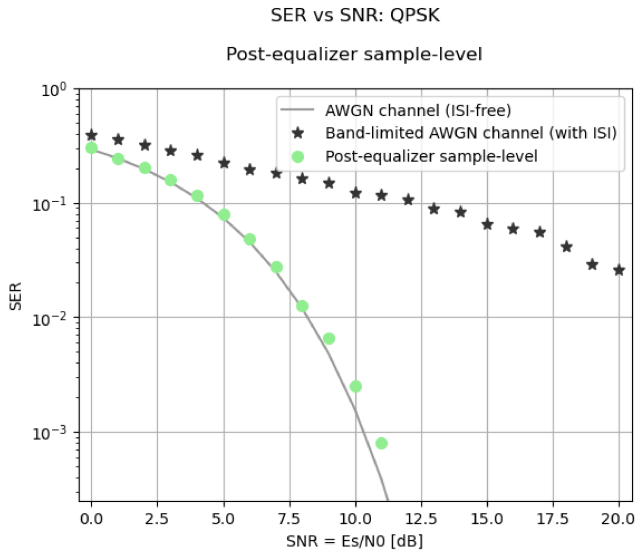


Figure 7.17: Post-equalizer at sample level.

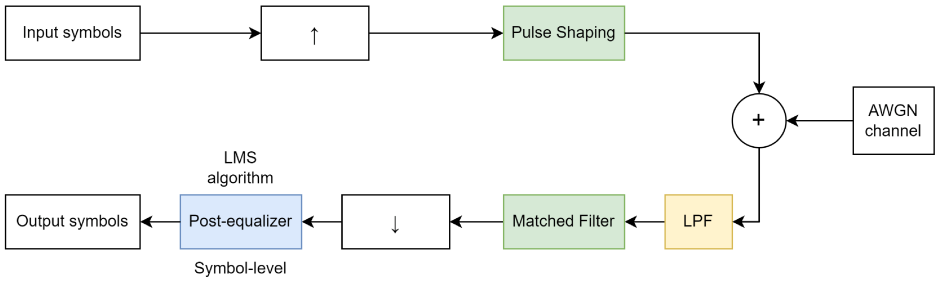


Figure 7.18: Block diagram of the post-equalizer at symbol-level.

AWGN channel without any compensation.
This is actually a drawback that could affect a post-equalizer implemented at symbol-level in channels with severe intersymbol interference [35]: due to the downsampling operation, we loose a big amount of information so that we are not able to ensure the channel equalization and recover for the impairments at the receiver as effectively as the sample-level equalization.

Now that we have presented the setting and the conventional equalization methods used to compensate for the channel penalties, in the next Chapter we will present the non-conventional ML-based framework we propose in this work.

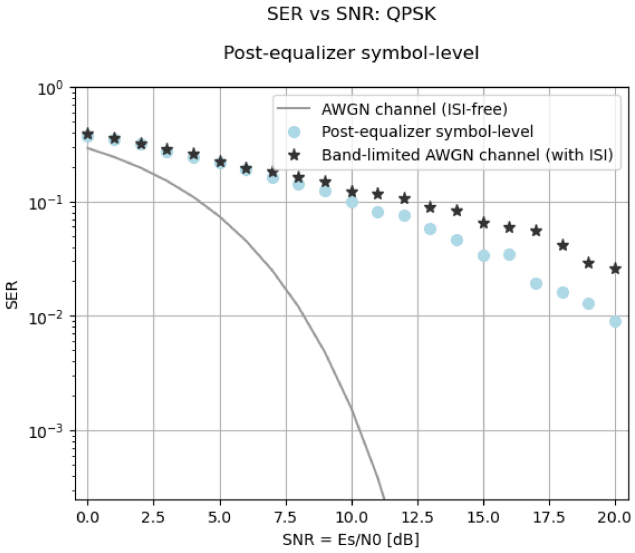


Figure 7.19: SER vs SNR: post-equalizer at symbol-level.

8

ML framework: implementation and discussion

In this work, we propose a ML-based framework aimed at providing equalization capabilities in a channel with impairments. The channel we used to test the proposed framework is the well-known band-limited AWGN channel, but, in principle, this preliminary study could be extended to more complex scenarios where the available equalization methods haven't achieved the optimal solutions yet and, thus, where ML could enable a performance improvement.
Starting from the structure we presented in Chapter 7, instead of performing the equalization via conventional techniques, we replaced the PS and/or MF blocks with trainable FIRs filters.
First, we will describe the design of the proposed ML model and explain the methodology we used to perform the training.
Afterwards, we will present the results we obtained expressed in terms of metrics like the SER and the Eye-diagrams. Then, we will compare them with the outcomes of the post-equalizers implementations in order to achieve our final goal: proving the effectiveness of the proposed ML-based framework and, eventually, validating it.

8.1 ML-based methodology

Since we are working on a Linear Time-Invariant (LTI) system affected by a LPF and by additive white Gaussian noise, we implemented a ML framework that can be formally defined as a linear regression model, capturing linear relationships between feature data and target data.

8.1.1 Problem outline: a Linear Regression Model

In this work, our communication system will be modelled as linear regression model in which we will learn the parameters to achieve the optimization the Pulse Shape to be injected into a band-limited AWGN channel.
Given the regression equation, the training is aimed at finding the linear dependence between two or more independent and dependent variables expressed quantitatively by the following mathematical expression:

$$\begin{aligned} \hat{\boldsymbol{y}} &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \\ &= \sum_{j=0}^{n} \theta_j x_j \\ &= \boldsymbol{\theta}^T \boldsymbol{x} \end{aligned} \tag{8.1}$$

Applied to our communication scenario, the vector $\boldsymbol{\theta}$ represents the FIR weights that we want to learn; the dependent variable $\hat{y}$ will be the estimate of the received symbols; $x_1, x_2, \ldots, x_n$, the n-dimensional independent variable represents the samples of the transmitted signal captured before the convolution with the FIR that we want to learn; finally, the multiple linear regression model aims at fitting the relationship between the received symbols $\hat{y}$ and the samples x of the signal. In the formal definition, θ_0 is the regression constant and $\theta_1, \theta_2, \ldots, \theta_n$ are the regression coefficients.
Finally, the Mean Squared Error will be our loss function $\mathcal{L}(\theta)$ that can be expressed as follows:

$$\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2 \tag{8.2}$$

where m is the cardinality of the set $\hat{y}$ that contains the estimated symbols at the receiver. Therefore, m represents the length of the message that we transmitted through the channel.

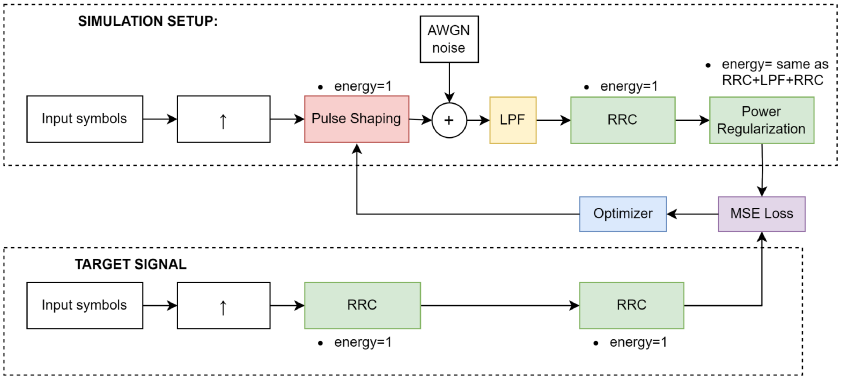
Three different scenarios

In communication systems, the PS is usually fixed to a well-known waveform, such as the RRC, and the system is provided with other blocks that implement strategies to compensate for distortion, like post-equalization.
In this work, the system will not be provided with equalizers, but the pulse shaper and/or the matched filter will be replaced by a linear model that we will train in order to learn the optimum PS and/or MF to be injected into the channel.
In other words, we want to learn the parameters of a FIR able to compensate for the distortion introduced by the band-limited AWGN channel. In the conventional AWGN channel scenario, there exists a well-known theory and the solution to avoid ISI can be found analytically. However, in this work, we are interested in a novel ML-based methodology to understand whether the linear model, i.e. one single FIR that we train, can replace both the Pulse Shaper and the Equalizer components.
With this aim, we will analyze three different scenarios:

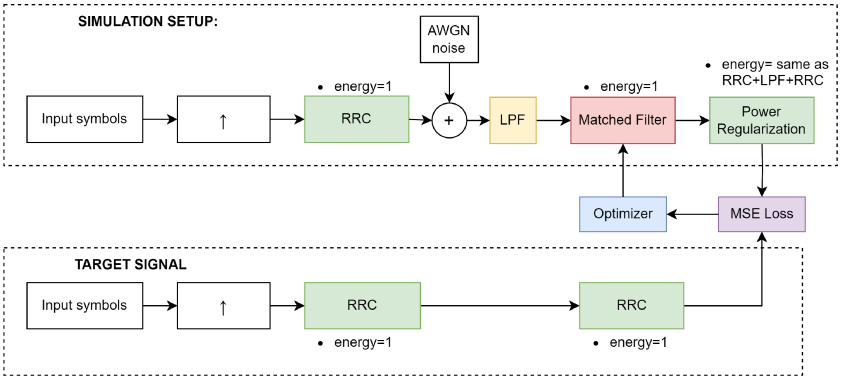
1. Learning the parameters at the transmitter with a RRC at the receiver;
2. Learning the parameters at the receiver with a RRC at the transmitter;
3. Learning the parameters at the transmitter and at the receiver jointy.

In Figure 8.1, we can see the setups for the ML-based framework of the three scenarios, that will be described thoroughly in the next paragraphs.

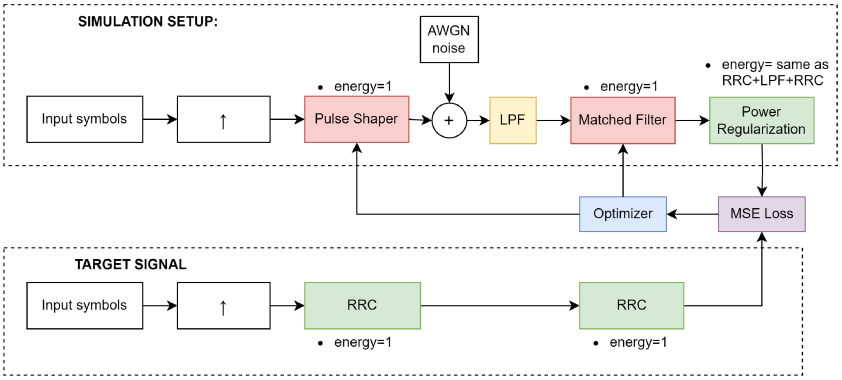
Energy normalization In Chapter 7 we normalized the energy of the RRC of the conventional AWGN channel simulator to 1. Similarly, a crucial aspect that was considered in the ML part was the energy normalization of the FIR weights to be learned. We introduced a normalization block in order to ensure that the learning weights have always an energy equal to 1 at each execution of the Forward function in our model. As a result, we will be consistent with the value of the SNR given to the model as an input and we will not alter it during the simulation. Without this normalization, we obtain outstanding performances at the receiver, but we are cheating the results because this is the consequence of an energy enhancement and not the result of an energy-efficient optimization of the parameters of the FIR filter.
In Figure 8.2 - that is relative to Scenario 1) - we can see that the FIR filter weights



(a) Learning the parameters at the Transmitter with a RRC at the Receiver.



(b) Learning the parameters at the Receiver with a RRC at the Transmitter.



(c) Learning the parameters at the Transmitter and at the Receiver jointly.

Figure 8.1: Block diagrams of the three scenarios.

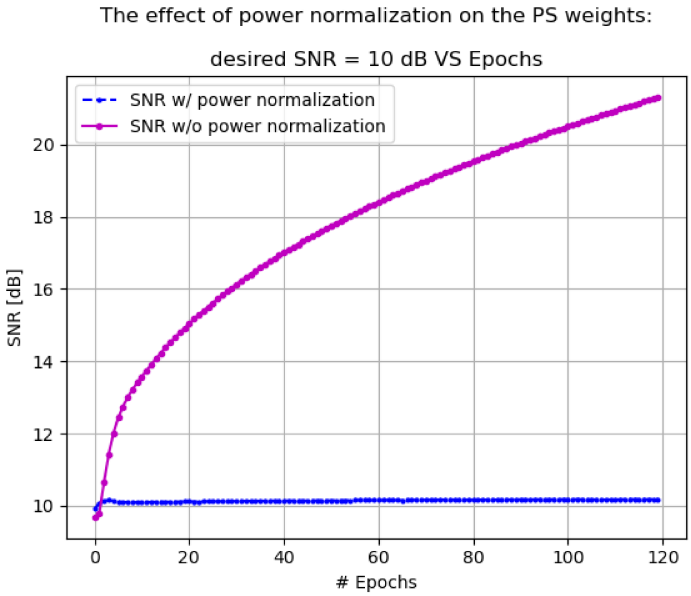


Figure 8.2: Exploring the effect of the power normalization block on TX weights and, thus, on the resulting SNR.

that we are training can increase their energy indefinitely if we do not set any constraint or bound and, thus, we are manipulating the SNR of the transmitted symbols. Whereas, we want the SNR to be a fixed parameter in the system. Therefore, in order to ensure this constraint, at each execution of the Forward function, we implemented a block that performs an energy normalization such that $\sum_{l=1}^{L_{RRC}} |w_l|^2 = 1$ where L_{RRC} is the length of I and Q channel FIR filters that we are training. Similar considerations can be discussed for scenarios 2) and 3).

Model design

The implementation of the model consists in the generation of a communication system with a band-limited AWGN channel, thus, we used the same blocks that we built for the traditional band-limited AWGN channel simulator we presented in Chapter 7 except for the PS and/or the MF, whose parameters are the weights that will be learned in the ML framework.
The input of the linear model is the symbol sequence $\{s_k\}_{k=1,..l}$ and the target output $\{s_k\}_{k=1,..l}$ is the sequence itself since at the receiver we want to receive the correct symbol that we sent over the channel after it went through several transformations. In addiction, the linear model is fed with other parameters that

we want to fix, that include the SNR at which the signal is transmitted and, thus, the power spectral density of the noise N_0 that affects the signal along the propagation, the number of symbols l that we want to simulate, the parameters that define the RRC such as the symbol period T_s and the roll-off β, etc. At the beginning, we initialized the parameters to random values taken from a Gaussian distribution with zero mean $\mu = 0$ and standard deviation $\sigma = 1$. Then, following suggestions of the authors of [32] - that encourage to take advantage of the channel knowledge to facilitate the learning process - we initialized the parameters of the Pulse Shaper and/or Matched Filter (depending on the considered scenario) to a conventional RRC.
After the definition of the model and the implementation of its Forward function with all the pre-defined blocks inherited from the conventional band-limited AWGN channel implementation, in order to implement the learning procedure, we set a loss criterion, e.g. the *MSELoss()* provided by the Pytorch Library, and we selected an optimizer. In our work, we implemented both a gradient-based and a gradient-free optimization algorithms that we described in Chapter 6. In a gradient-based method, the update of the parameters of the trainable FIR is made possible by the well-known SGD optimizer or by Adam optimizer that rely on the Backpropagation algorithm to compute the gradient. Whereas, in gradient-free methods, we selected the SPSA stochastic optimizer that uses an approximation of the Gradient to update the parameters.

Performance of gradient-based and gradient-free optimizers in the considered scenario

In this section, we will do some considerations about Adam algorithm, as representative of the gradient-based optimizers, and SPSA algorithm, that is instead a gradient-free optimizer.
As we discussed in Chapter 6, gradient-free optimizers can be essential in use cases in which it is not possible to compute the derivative of the loss function $\mathcal{L}(\cdot)$, for instance when it is too complex to calculate or when it does not exist.
In the considered case of the band-limited AWGN channel, there are no issues in the computation of the derivative of the loss function with respect to the FIR parameters that we want to learn and, thus, a gradient-free method would not be necessary. However, in the view of extending this work to more complex scenarios, such as the short-reach optical link we presented in Chapter 2, we

decided to implement also a gradient-free optimizer, i.e. SPSA. The algorithm was very sensitive to the choice of hyperparameters whose selection was more difficult than in the case of Adam, where we just need to set the learning rate. Finally, following the suggestions we found in [46] about the initialization of the hyperparameters (a, c, A, α, and γ that we presented in Chapter 6), the implementation of SPSA succeeded and the algorithm was able to converge.
Assuming to evaluate the complexity of an algorithm with the number of iterations needed to reach the convergence, the SPSA algorithm took much more iterations than the considered gradient-based algorithm (in this case Adam).
In case in which the computation of the derivative has a high computational complexity, using SPSA is convenient as it approximates the derivative instead of computing it. However, since it relies on an approximation, it will take more iterations to converge. In the context of a band-limited AWGN channel, avoiding the computation of the derivative is not advantageous and, thus, in the considered case, we only experience the drawback of having more iterations to reach the convergence without any improvement in terms of the cost of a single iteration since the derivative itself is not so complex to compute.
As a proof of concept, in Figure 8.3, we plotted the MSE loss computed at the end of each batch for both Adam (in orange) and SPSA (in blue) optimizers that we obtained for Scenario 1) (learning the parameters at the transmitter). These are the results of specific simulations where the model was fed with input datasets of the same size; also the number of epochs and the batch size were the same in the two implementations. In the considered example, Adam converges quite fast, after a few hundred of iterations, while SPSA reaches the convergence after the 3000-th iteration, continuing to exhibit a slightly oscillatory and noisy behaviour.
To sum up, we selected the same simulation parameters except the optimizer and then we compared the obtained MSE loss. To prove the higher complexity of SPSA algorithm in the considered case, we have shown that the SPSA MSE loss is noisier and takes more time than the Adam MSE loss to converge.
Given the limited computational resources we had, since in our specific case it is not strictly necessary to use a gradient-free optimizer, for further analysis, we decided to use only gradient-based optimizers like Adam and also SGD, that led to similar results in terms of complexity.

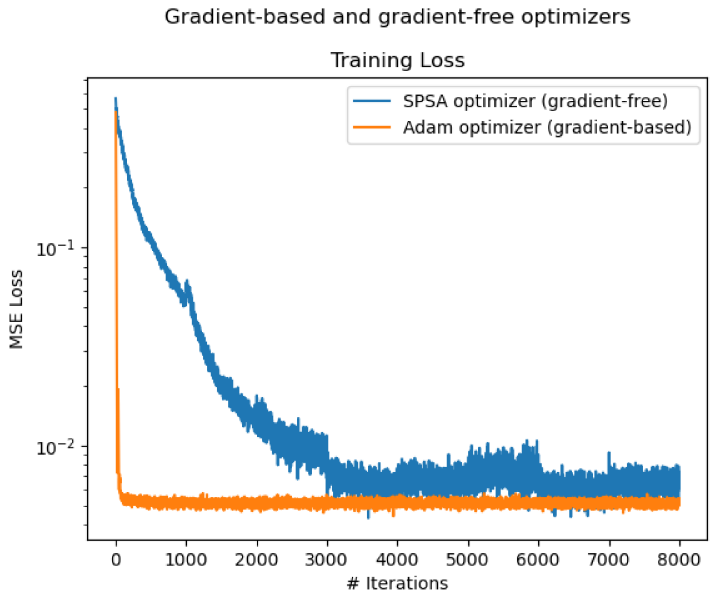


Figure 8.3: MSE loss of Adam (gradient-based) and SPSA (gradient-free) optimizers.

8.1.2 Hyperparameters optimization and model training

As soon as we defined both the Initialization and the Forward function of the linear model, we proceeded with the implementation of the learning process. We fixed many parameters in order to create the framework for the learning task: we set the size l of the input data $\{s_k\}_{k=1,\dots l}$, the number of epochs - i.e. the number of times that the generated dataset is fed to the model - and the batch size - the portion of the data that has to be considered in a single iteration within a certain epoch.
The structure was built with nested *for loop* in which we slid throughout the epochs and then throughout the batches. This framework allowed us to perform the training of the model. In addiction, in order to make an appropriate choice of parameters and hyperparameters such as the size of the input dataset or the learning rate of the selected optimizer, we also performed the validation of the model using another generated dataset: within the framework, at the end of each epoch, we computed the loss function for both the training set and the validation set, or, in other words, the training and the validation loss respectively and we checked that they matched each other to ensure that the training was done properly without underfitting or overfitting (as we discussed in Chapter 6). As an example, in Figure 8.4, we plotted the training and the validation

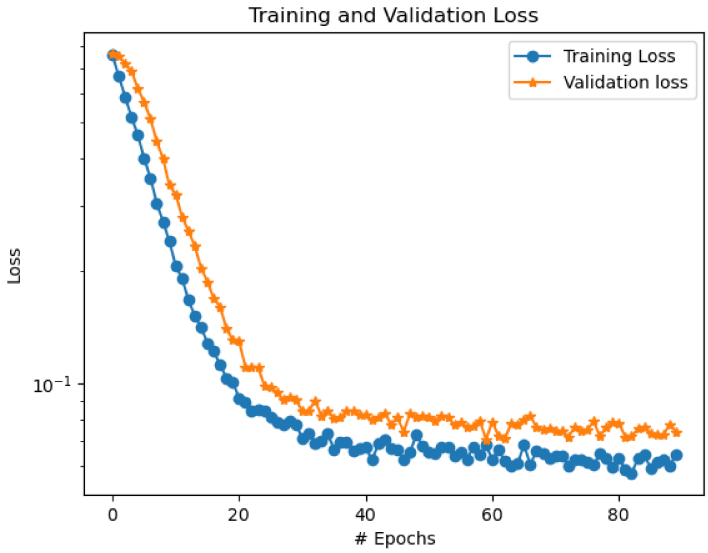


Figure 8.4: Training and Validation Loss (Adam optimizer).

loss that we computed at the end of each epoch during a specific simulation in which we used Adam optimizer.
After that the training and the validation were completed, finally, we randomly generated another dataset with other symbols and we tested the learned parameters in a new framework and verified that the training error matched the test error. Besides, to have a better understanding of the performance of the system, we also computed the SER at the end of the test to check whether the learning was effective and, thus, if the learned parameters were able to compensate for the channel distortion.

To sum up, first we prepared the environment for the simulation: we designed the linear model, we generated the input data (i.e. the QPSK symbols) and we initialized the model to learn the parameters of interest. Then, we trained the model and meanwhile we performed the validation; finally, we tested the FIR weights we obtained with the training.
Moreover, we computed the SER at the end of the communication system implemented with the learned parameters and then we compared the obtained SER with the SER of the AWGN channel and with the SER of the band-limited AWGN channel with two RRC, one placed at the transmitter and one at the receiver. We implemented the setup we have just described for all the three different cases that we presented in the previous section.

Strategies and methods: In this paragraph, we will describe more in detail the strategies that we used to train our model within the simulations.
At the beginning, the learning of the parameters was performed on an artificial channel characterized by the absence of the AWGN and by the only presence of the Bessel filter. In this simulation, we were able to compensate completely for the penalty introduced by the LPF, i.e. converging to a zero-forcing approach.
After this preliminary simulation with a noiseless band-limited channel, we gradually started to introduce the noise. We adopted the following strategy: at the beginning of a simulation with a new SNR value, the weights of the filter to be trained were initialized to the values of the parameters that were learned in the previous iteration. We started from the case SNR = 40 dB and then we iteratively decreased the value of the SNR of a step of 5 dB. Therefore, we simulated the cases with SNR equal to 35 dB, 30 dB, 25 dB and 20 dB. As soon as we reached the case SNR = 20 dB, we started proceeding slowly, training the model decreasing the SNR of one unit until we reached 0 dB.
This approach was suggested by the authors of [15], that state that, in the context of ML applied to Digital Communications, it is convenient to use the preliminary knowledge about the communication system to improve the performance of the model.

8.2 SER PERFORMANCE ANALYSIS

In this section, we will describe and evaluate the computed SER for each of the three scenarios individually.

LEARNING THE PARAMETERS AT THE TX

In the first scenario, the aim is to learn the parameters of the PS in order to compensate for the channel impairments. At the receiver we set a conventional RRC filter.
The trainable FIR that we want to learn should be the convolution between the RRC filter and a pre-distortion block able to invert the distortion introduced by the channel. In Figure 8.5, we can see the SER vs SNR curve in this context: the training is able to mitigate the distortion introduced by the channel, but there is a remaining gap with respect to the AWGN-only case. In this scenario the ML model should be able to learn the convolution of two filters in one shot. We

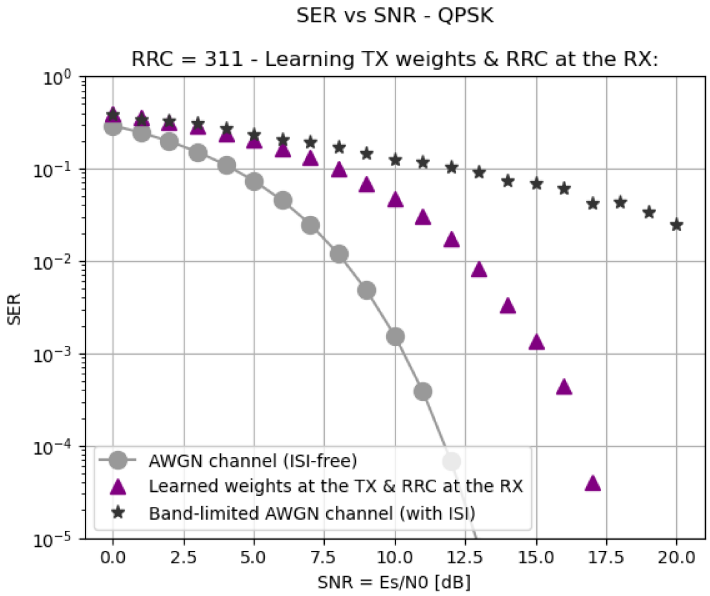


Figure 8.5: Learning the parameters at the TX and RRC at the RX.

adopted the strategy described in the previous section, with a gradual decrease of the SNR. This led to performance improvements with respect to the case in which we initialized randomly the FIR taps and even compared to the case in which we initialized the taps with the conventional RRC. However, the outcomes are still far from the Shannon bound, i.e. from a complete compensation.

Learning the parameters at the RX

The second scenario consists in learning the parameters of the MF at the receiver side. At the transmitter side, we are using a traditional RRC. Observing Figure 8.6, we can notice that the learning process is effective and we are able to learn the parameters to compensate for the penalty introduced by the channel almost completely. Referring to the conventional setup, in this scenario, the trainable FIR that we want to learn corresponds to the convolution of the RRC matched with the PS and the post-equalizer.

Learning the parameters at the TX and the RX jointly

In the last scenario, we learned the parameters of the PS and the MF at the transmitter and the receiver jointly. This approach is called end-to-end optimization and is a novel technique that we found quite often in recent works (as we discussed in Chapter 3).

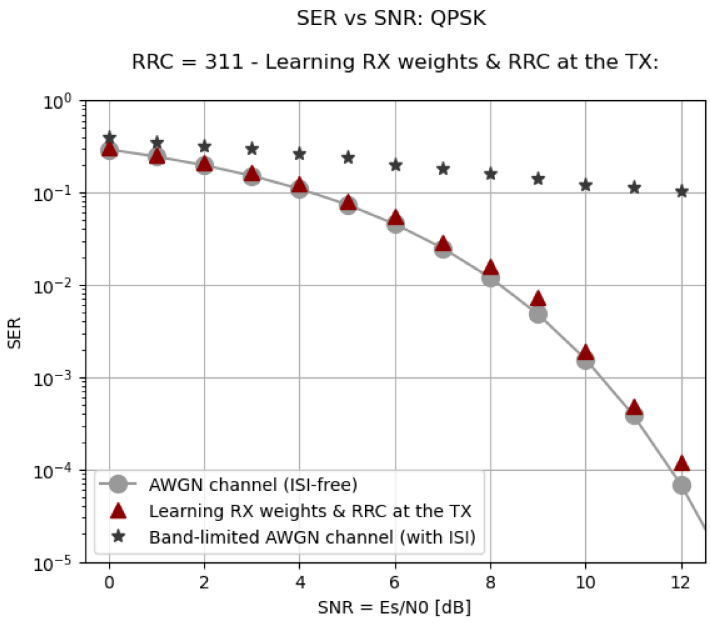


Figure 8.6: Learning the parameters at the RX and RRC at the TX.

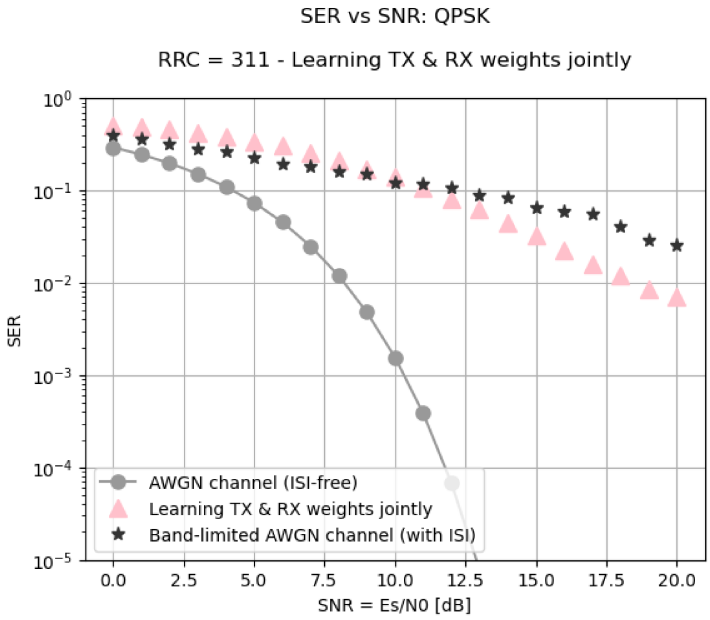


Figure 8.7: Learning the parameters at the TX and at the RX jointly.

The obtained SER is shown in Figure 8.7. Observing the Figure, we can see that for low SNR values the effect of the learned parameters is even worsening the performance that the system would have had without any compensation strategy. Whereas, as soon as the SNR increases, an improvement can be noticed, but anyway the performance is the worst among the three considered scenarios. In this scenario, we are trying to learn the convolution of three filters in one single step - thinking about a traditional implementation, two RRCs and a post-equalizer split between the transmitter and the receiver - one of which is placed at the transmitter where there is no channel knowledge and this task seems more challenging to be performed.

8.3 Analysis of the three scenarios

In this section, we will describe more in detail the results we achieved in the three scenarios showing the obtained FIR parameters in time domain and in frequency domain for several SNR values and also Eye-diagrams at SNR = 40 dB. Furthermore, we will compare the three ML outcomes between each other.
The best result and actually the only one that is able to compensate completely for the channel impairment is Scenario 2), where the learned FIR is placed at the receiver and it acts as the convolution between the MF and the post-equalizer. Its performance matches the performance of the post-equalizer implemented at sample-level, with the difference that it incorporates into one single block the two components of the conventional setup. Whereas, the performances of the other two scenarios - learning the parameters at the transmitter (Scenario 1) and at the transmitter and at the receiver jointly (Scenario 3) - are quite far from the AWGN channel SER curve. In Figure 8.10, 8.11 and Figure 8.12, we can see the shape of the pulses (time domain) and their magnitude responses (frequency domain) captured at the receiver, thus they are the results of the convolution between the PS, the LPF and the MF for all the three considered scenarios, in time and frequency domain respectively.
Let us compare these plots with the ideal RC that we would have obtained if we did not have the LPF in between.
In Scenario 1), observing the time domain plot, the learned waveforms are quite similar between each other and, decreasing the SNR, we can notice a broadening effect in which the peak exhibits a slightly lower energy and, thus, the energy of the pulse - normalized to be equal to 1 - is spread on a larger time interval.

Let us recall the Nyquist criterion: it says that there must be zero crossings in correspondence of all the peaks adjacent to the considered symbol to avoid ISI. Looking at the time domain plot (Figure 8.10a), we can notice that in this case the RC's and the learned waveforms zero crossings are not matched. This results into a poor performance in terms of SER that proves that the learned waveforms are weakly able to mitigate the effect of ISI. Observing the frequency domain plot (Figure 8.10b), going towards lower SNR values, we can see that the highest frequencies exhibit a progressively lower amplification. Finally, comparing them with the RC spectrum, in general, we can notice that the spectrum of the learned waveforms lacks the high frequency components that would have allowed to reproduce the original RC pulse shape. In addiction, observing the Eye-diagram in Figure 8.9a - even though it has a shape that recalls the shape of the target one (Figure 8.8) - we can notice that the learned waveform is not accurate and the ISI is still present. Indeed, the ISI implies the reduction of the opening of the eye and hence the detector will be able to recover the transmitted symbol only in presence of a small amount of addictive noise. Moreover, the ISI is also responsible for the distortion of the position of the zero crossings that cause the system to be sensitive to synchronization errors [35].
Scenario 2) (Figure 8.11) is the only one that provides an almost perfect compensation of the signal. However, as we can see from Figure 8.11a, the learned waveforms do not match perfectly with the original RC waveform and they exhibit different shapes depending on the SNR value: we notice that the learned spectrum (Figure 8.11b) implements the amplification of the high frequency components; this amplification increases with the decrease of the SNR. Such condition is similar to what we experienced with the post-equalizers' setup: at the beginning, we implemented a zero-forcing equalizer that was able to invert the channel frequency response with the collateral effect of enhancing the noise (remember Figure 7.16). Similarly, the learned spectrum in Scenario 2) is such that the high frequencies have a larger magnitude as the SNR decreases. Comparing the Eye-diagram (Figure 8.9b) with the target one (Figure 8.8), we can state that the two shapes match each other. Even though the learned waveform is not as accurate as the target, it is accurate enough to ensure an almost perfect compensation for the channel impairments as we saw in Figure 8.6.
Differently from the other Scenarios, in Scenario 3) (Figure 8.12), we can notice that the learned waveforms are far from the ideal RC: instead of having a fast decay of the pulse - that is generally good to reduce the intersymbol interference

- they exhibit several ripples. As a result, the available energy is more spread along the taps instead of being concentrated on the main peak. Furthermore, the learned waveforms exhibit several zero crossings that are unfortunately not well-tuned as there is still a lot of interference between symbols at the receiver and, thus, a high SER, that is the worst among the three considered Scenarios. Looking at the spectrum of the learned waveforms (Figure 8.12b), we can realize that they are quite far from the desired one: the high frequency components should be much more enhanced to mimic the RC spectrum. Moreover, we can notice that the lower the SNR, the lower the amplification of the high frequency components. For instance, when SNR = 0 dB, the learned spectrum shows that it does not have high-pass filter properties, while it amplifies low frequency components and, thus, worsens the distortion of the signal. Indeed, remembering the SER curve in Figure 8.7, at low SNR, Scenario 3) showed a worse performance than the band-limited AWGN channel with RRCs at the Transmitter and at the Receiver. Indeed, in the time domain, with the decrease of the SNR, we can see that the learned waveform ripples are more evident: they carry more energy that is "subtracted" from the main peak (that is hence slightly smaller) and that affects the adjacent symbols. As a result, we experience a worsening of the performances and, thus, a higher SER. For what concerns the Eye-diagram (Figure 8.9c), we can notice that the waveform's shape is different from the shapes that we see in the other scenarios: as we said, it exhibits different ripples and zero crossings, that apparently do not coincide with the zero crossings that we should have if the Nyquist criterion was met given the bad performance in terms of SER that we obtained in this scenario. To sum up, in Scenario 1) and Scenario 3), the proposed ML framework mitigates only partially for the channel impairment. Indeed, the SER exhibits only a weak improvement, except for low SNR values in Scenario 3) where the training even worsens the performance. Between these two scenarios, the learning at the TX (Scenario 1) exhibits a slightly better performance than the end-to-end learning scenario (Scenario 3). The evaluation of the Eye-diagrams in all the three scenarios asserted this analysis.

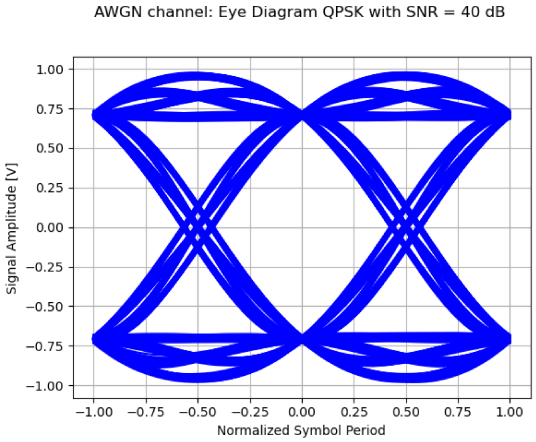
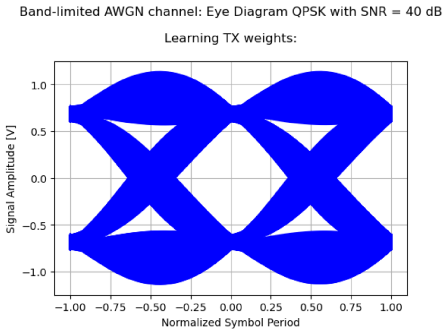
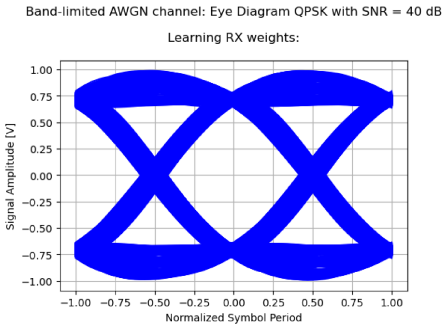


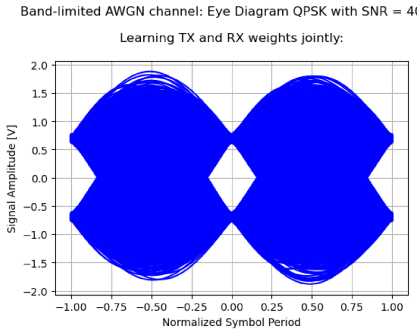
Figure 8.8: Target Eye-diagram: ISI-free AWGN.



(a) Learning the parameters at the TX.

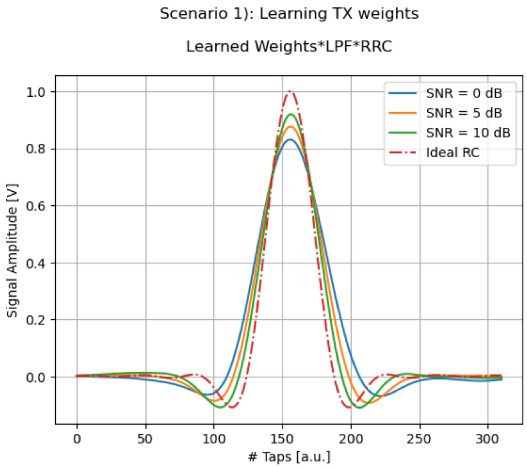


(b) Learning the parameters at the RX.

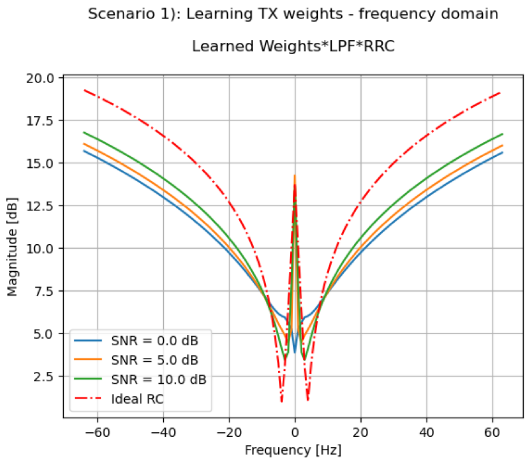


(c) Learning parameters at the TX and RX jointly.

Figure 8.9: Eye-diagrams at SNR = 40 dB in the three scenarios.

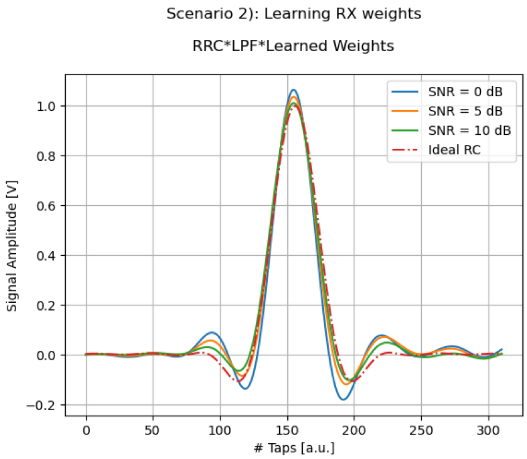


(a) Time domain.

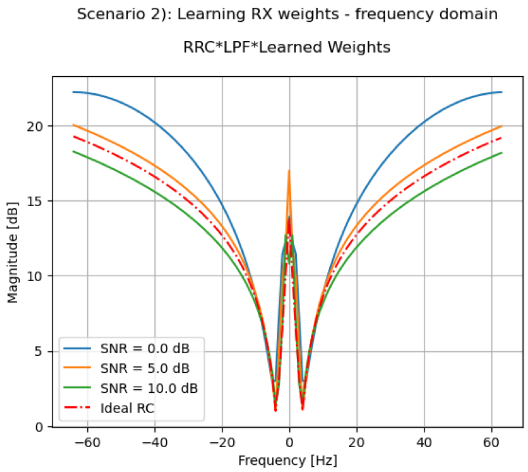


(b) Frequency domain.

Figure 8.10: Scenario 1): Learned weights at the TX - time and frequency domains.

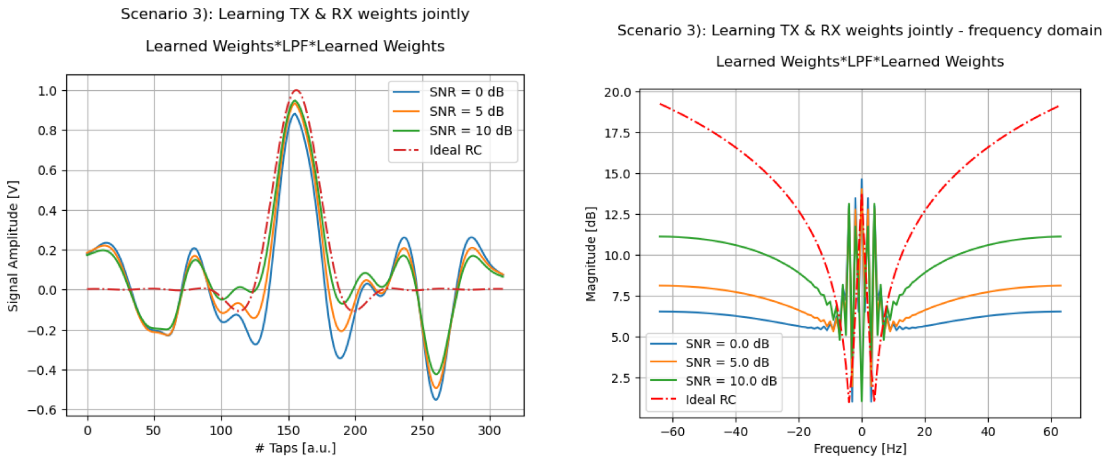


(a) Time domain.



(b) Frequency domain.

Figure 8.11: Scenario 2): Learned weights at the RX - time and frequency domains.



(a) Time domain.

(b) Frequency domain.

Figure 8.12: Scenario 3): Learned weights at the TX and RX jointly - time and frequency domains.

8.4 Comparison between the ML-based framework and the conventional setup

In this section, we compare the performance of the non-conventional proposed ML-based framework with the two more traditional frameworks with post-equalizers - the first at sample-level and the second at symbol-level - with the aim of providing a benchmark to validate our results. Moreover, we will try to make hypothesis and give a possible explanation to the performance obtained in the three scenarios, trying to have an insight on how does the learning process work. In Figure 8.13, we plotted all the considered SER curves to facilitate the comparison.
The achieved performances will be compared as follows:

- Scenario 2) with the post-equalizer at sample-level;
- Scenario 2) with Scenario 1) and Scenario 3);
- Scenario 3) with the post-equalizer at symbol-level.

Scenario 2) with the post-equalizer at sample-level. In Figure 8.14, we can see the obtained SER curves of Scenario 2) and of the sample level post-equalizer. Observing the Figure, we can see that the two curves match each other and,

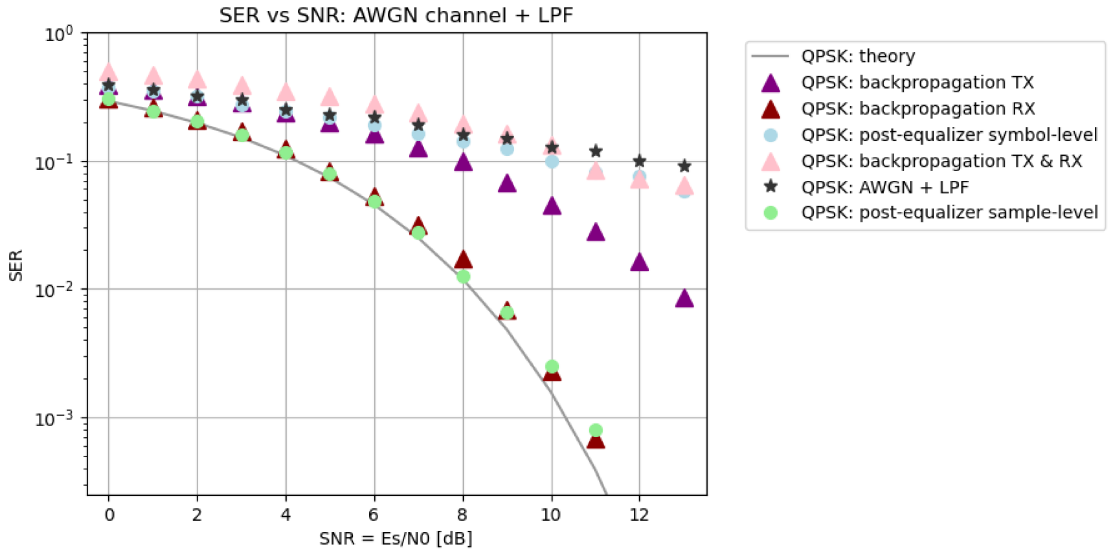


Figure 8.13: SER vs SNR: the three scenarios and the post-equalizers.

moreover, they are able to provide an almost complete compensation that approaches the theoretical curve.
The post-equalizer is learning the parameters that invert the channel impairment, while Scenario 2) does not have a conventional RRC at the receiver and, thus, the trainable FIR is supposed to implement the convolution of two filters, i.e. the RRC and the post-equalizer, in order to achieve the compensation for the penalties.
This two cases are quite similar to the extent that the training of the FIR weights rely only on the channel output and, thus, the channel is not even needed.

Scenario 2) with Scenario 1) and Scenario 3). Let us go on with the second analysis. In Scenario 1) (learning the TX weights), the FIR filter should be able to learn the weights of the filter that result from the convolution between a RRC and a pre-distortion block. Given that the performance of Scenario 1) is much worse than Scenario 2), we can try to think about the learning process in Scenario 2) to provide an intuitive explanation. The FIR filter is the last block of the communication system and, thus, the variation of the parameters has a direct impact on the predicted values. Whereas, in Scenario 1), the TX FIR parameters that we are learning are processed by all the blocks of the communication system: they are affected by the AWGN, then pass through the LPF and finally the signal is convolved with the RRC at the Receiver. Then, we compute the MSE that is used by the optimizer to update the TX weights.

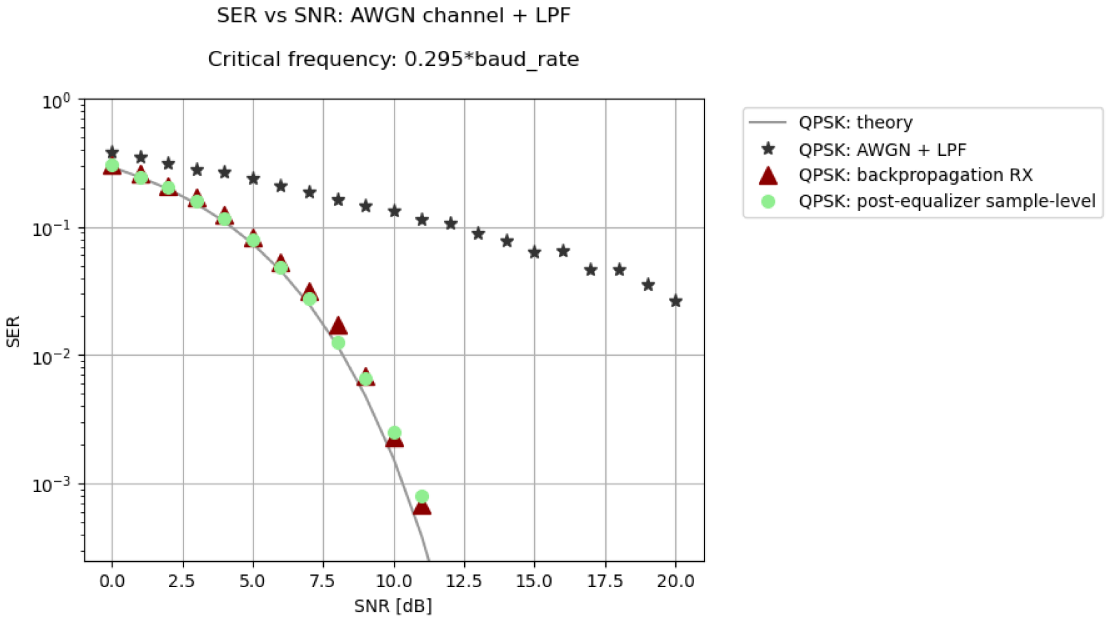


Figure 8.14: Comparison between Scenario 2) and post-equalizer sample-level.

Differently from Scenario 2) (learning RX weights) where the predicted values coincide with the output of the RX FIR filter, in this case the predicted values are the result of several transformations (that include the channel, not known at the transmitter side) that affect the trainable TX FIR weights. To sum up, compared to Scenario 1), in Scenario 2) we are training the FIR weights at the last stage of the communication system and, thus, we do not even need the channel knowledge to perform this operation.
Whereas, in Scenario 1), the trainable FIR weights are placed at the transmitter, thus, before that the channel effects have occurred. Thinking about a traditional implementation, it means that the training should be able to learn the convolution of the RRC and of a pre-distortion block placed at the transmitter and, thus, it should be able to predict the FIR parameters without knowing the channel characteristic. Apparently this resulted into a more challenging learning task.
Similar considerations can be done when we compare Scenario 2) with Scenario 3). Scenario 3) (learning TX and RX weights jointly) corresponds to a scenario in which we are trying to learn jointly the weights of a RC and a post-equalizer split between the Transmitter and the Receiver. It is the so-called end-to-end learning and it seems to be even more challenging than Scenario 1) as we can suppose if we think about how the learning process works when the trainable FIR parameters are both at the TX and at the RX and, in addiction, the channel

characteristic is unknown.

Scenario 3) with the post-equalizer at symbol-level. With this analysis, our aim is that of providing a validation of the proposed ML-based setup. Looking at the SER that we obtained in Figure 8.13, as we discussed, we can see that the post-equalizer at symbol-level is barely able to compensate for the combined effect of the noise and of the bandwidth limitation of the channel. The more traditional implementation of the post-equalizer provides us a benchmark, i.e. the post-equalizer at symbol-level can help us to interpret and to give an explanation to our results.
In the considered symbol-level post-equalizer setup, we achieved only a weak mitigation of the penalties induced by the combination of the LPF and the AWGN. As we discussed in Chapter 8, if the communication channel is affected by severe intersymbol interference, a symbol-level post-equalizer might not be sufficient to ensure the channel equalization as well as the sample-level equalization does [35].
Indeed, due to the downsampling operation, we loose a big amount of information and this could be responsible of the fact that we are not able to recover for the impairments at the receiver.
The obtained results with the symbol-level post-equalizer can help us to provide a possible explanation to the outcomes of the proposed ML-framework, increasing their interpretability. We can compare the SER we got in the post-equalizer symbol-level implementation with the SER of Scenario 3). As we can see in Figure 8.15, Scenario 3) of the ML framework exhibits a SER that is comparable to the symbol-level post-equalizer SER. Let us describe the two curves more specifically. We can notice that the symbol-level equalizer is able to provide a very weak mitigation of the performance at any SNR value. Whereas, Scenario 3) has a slightly different behaviour. For low SNR values (more precisely from 0 dB up to 10 dB), the learned FIR parameters even worsen the channel's distortion, resulting in a SER that is higher than the SER we have without implementing any equalization method. Within the learning process, in such scenario, the distortion is so severe that the learning overfits it, enhancing the noise, even more than what happens in the post-equalizer symbol level scenario. Going on with the analysis, considering higher SNR values (starting from 11 dB), we can notice that the post-equalizer and Scenario 3) SER exhibits a similar trend, with the ML-based framework SER slightly better than the results that the post-equalizer

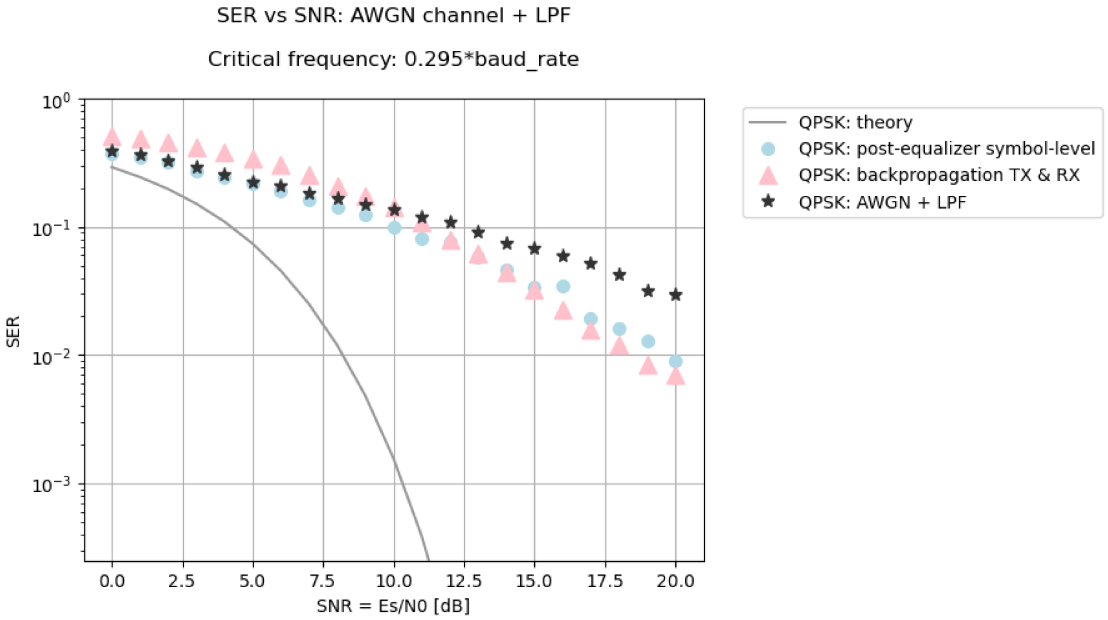


Figure 8.15: Comparison between symbol-level post-equalizer and Scenario 3) (learning TX and RX jointly).

is able to achieve.

To sum up, assuming a high-level view, we can see that the two curves present a similar behaviour. Looking more into the details, we have noticed the differences between the two curves, that may find a possible explanation to the fact that we are learning from data and the ML-based framework reflects the data that feeds the model: at low SNR, the performance are worse, while increasing the SNR we can see that ML-based framework slightly outperforms the symbol-level post-equalizer. With the post-equalizer implementation, we obtained a benchmark and, thus, we were able to propose an interpretation to the ML framework results.

8.5 Final considerations

To conclude, in our setup we explored the effects of the variation of many parameters. We varied the model input parameters (such as the filters length, both RRC and LPF, the roll-off factor β, the SNR values, etc.), the parameters of the ML framework (input size, number of epochs, batch size, etc.), the optimizers (Adam, SGD or SPSA) and the hyperparameters like the rate and the momentum of the selected learning algorithms. The tuning of such parameters

was a time-consuming task. At the beginning it was performed "at hand", but then we implemented it with the *GridSearch* algorithm to tune the parameters more systematically.
A parameter that we maintained for every setup we simulated is the loss function, i.e. MSE. In future works, other frameworks can be explored, for instance making a different choice of the loss function, given that ML is very sensitive to the choice of parameters, hyperparameters and loss function used in the simulations [44].

9

Conclusions and Future Work

In this work, we tested and validated a non-conventional ML-based framework aimed at the equalization of a channel with impairments. For the goal of this work, the channel that we selected was the band-limited AWGN channel, thus, a simple source of impairment, that allowed us to easily implement more conventional equalization methods - such as post-equalizers - to benchmark and validate the proposed ML-based framework.
The first step was the implementation and the validation of a traditional AWGN channel simulator, provided with a RRC Pulse Shape filter at the transmitter and its corresponding Matched Filter at the Receiver. Then, as a source of impairment, we added a Bessel LPF that introduced distortion into the channel, causing intersymbol interference. Instead of adopting traditional equalization methods (like FFE or DFE), in order to compensate for the ISI, we proposed a non-conventional framework in which we replaced the two RRC filters with trainable FIRs. We proposed three setups: in the first we had one trainable FIR placed at transmitter, in the second the trainable FIR was at the receiver and in the third we had two FIRs one at the transmitter and one at the receiver that we trained jointly, in the so-called end-to-end learning approach, quite popular in recent literature. The training procedure aimed to learn the optimum pulse shapes - i.e. the proper taps for the trainable FIRs in practice - that ensure a correct transmission and reception, given the non-idealities of the band-limited AWGN channel.
In order to provide a benchmark for the proposed framework, we implemented a more conventional setup with post-equalizers. We realized two versions, one

at sample-level and another at symbol-level. The latter was done to cope and to be consistent with further use cases where number of samples per symbol is limited and where the equalization usually occurs at symbol-level.
In order to compare the obtained results, we relied on:

- quantitative metrics such as the MSE, the loss function we used for the training, and the SER, that we computed after the detection of the received signal;
- qualitative metrics, such as the Eye-diagram and the constellation diagram, that allowed us to display the signal, providing insight into its characteristics and behaviour.

In this way, we were able to analyze the obtained outcomes and to compare the performances we achieved in the three different proposed scenarios with the two setups with conventional post-equalizers. As a result, we achieved the validation of the ML-based proposed framework, that was the main objective of this work.
Tout court, in this work, we performed a methodological evaluation of a non-conventional equalization technique whose novelty consists in the following aspects and their interaction:

- the focus on Pulse Shaping and Matched Filter blocks;
- the application of ML-based techniques in the context of Digital Communication;
- the use of such innovative ML techniques to optimize the pulse shaping, providing our framework with trainable filters able to learn a proper pulse shape to mitigate the channel impairments.

More specifically, we took advantage of some of the recent techniques and practical strategies used in recent research papers about ML for Communications that we consulted within the literature review.
For instance, we adopted the so-called end-to-end training, a popular strategy that consists in the joint optimization of the transceivers. Applied to our specific scenario, we used this technique to train the transmitter and the receiver pulse shaping filters jointly. It represents a step forward compared to the traditional learning technique in which a block-based optimization is performed and, thus, where each block is optimized individually.
Another practical strategy we adopted is connected with the specific context of Communications where there aren't well-established practical methods on how

to perform the training yet. We tested our scenario on different SNR values starting from high SNRs (more specifically 40 dB) and then we gradually decreased it until we reached 0 dB. At every new iteration, we initialized the parameters of the trainable FIRs with the outcomes of the previous iteration to simplify the learning process. Thus, we tried to incorporate the domain knowledge into the training process with the aim of improving our results, following suggestions we found in recent research papers [15].
In general, in this work we wanted to follow the novel path that promotes the application of ML to the context of Communications.
As we discussed, the use of ML in this field is a quite novel research direction. The goal is to exploit the vast expert knowledge in ML with the long-term aim to incorporate Artificial Intelligence to Communication Systems, making them more robust and versatile. In addiction, among the various use cases that we analyzed, this approach can be particularly useful for the following Communication's scenarios:

- scenarios that have a known optimal solution, but that rely on an algorithm that is too complex for a real-time implementation;
- scenarios that still lack a proper model definition and for which the optimal solution is hence still unknown;
- scenarios for which - despite the availability of a mathematical model - no perfect analytical solutions are available.

This latter case includes the short-reach optical link that motivated this work. In such context, ML can be an important tool to achieve performance enhancements, meanwhile considering carefully the complexity of the solution that is the most critical factor in the specific case of short-reach optical links.
In this work, because of the limited amount of time, we tested and validated only the band-limited AWGN channel, that does not lie into the aforementioned use cases in which ML can provide an effective improvement since there already exist well-established techniques and methods to obtain the optimal performance whose bounds are well-known.
However, in order to incorporate ML in Communication Systems, the essential starting point is to test how does ML perform in well-known contexts and scenarios. In principle, ML has an outstanding potential as it has already been proved in other fields like computer vision or natural language processing [16].
Whereas, Communications is another story and, in the view of a more important role of ML in Communications, before adopting it on more complex scenarios

in which we could achieve effective improvements, as a preliminary stage, its applicability has to be tested and validated in scenarios in which we have an overall knowledge that allows us to increase the interpretability of our results and to perform a proper validation of the ML-based method comparing it with well-established traditional solutions.
This is exactly what we did in this work: we tested and validated a ML-based framework in the well known context of the band-limited AWGN channel.
Thanks to the validation process, the developed framework can be considered as a general-purpose structure that can be applied to more complex scenarios - for instance to channels affected by nonlinear impairments like the short-reach optical link - provided that it has been proved to be a valid alternative to more conventional methods in a well-known Communication scenario. In such use cases, in principle ML could enable the opportunity to improve performances and even to reduce the complexity of the systems, leading to the development of a novel generation of Communication Systems where it would play an essential role.

Future work In this final section, we will mention some straightforward and even more further extension for this preliminary work.
As we discussed, in this work, we explored the opportunities that a ML could offer in the field of communication focusing on the specific block of Pulse Shaping/Match Filter blocks with the goal of achieving the equalization of the channel impairments.
The first step that we have already anticipated and that actually motivated this work is the possibility to extend this ML-based framework to scenarios in which ML would enable an effective performance improvement like the short-reach optical channel. The adoption of an optical channel model would produce a more realistic scenario that would enable the setting of new parameters for the simulations. For instance, we could include parameters connected to the physical nature of the Optical Fiber and its deployment, such as the Chromatic Dispersion β_2, the attenuation, and also the fiber length to consider the losses that occur when the signal propagates along the transmission medium. Along with a characterization of the optical channel, with the aim of making our setting more realistic, we can also simulate the presence of other hardware devices - optical or electronics.
For instance at the receiver we could add a photodetector, whose effect paired

with the Chromatic Dispersion β_2 would result into nonlinear effects. To address the nonlinearity issue, on the path of works we found in the literature review, we could implement an autoencoder based on a NN would provide further insight into the Pulse Shaping and Match Filtering blocks that we started exploring in this work. Of course, the described scenario would need a novel implementation with new design choices, from the loss function (Mutual Information can be considered as an alternative choice) to the tuning of the hyperparameters that would require new thorough analysis and considerations.
Furthermore, in this work, we focused only on the proper delivery of data and on the equalization capability of the developed ML-based framework, evaluating metrics such as the SER. However, the role of PS/MF blocks is also that of limiting the bandwidth occupied by the signal.
In this view, it is possible to train the model to learn the optimal pulse shape aimed at the optimization of the occupied bandwidth. To achieve this purpose, other metrics should be introduced such as the SE that would allow us to find a trade off between the data rate and the occupied bandwidth. Furthermore, the single channel scenario could be extended to a multi-channel scenario with the implementation of Wavelength Division Multiplexing (WDM). A possible optimization would be aimed at reducing the Interchannel Interference (ICI) for which the already adopted SER could be a useful metric.

References

[1] G.P. Agrawal. *Fiber-Optic Communication Systems*. Wiley Series in Microwave and Optical Engineering. Wiley, 2012. ISBN: 9780470922828. URL: https://books.google.it/books?id=yGQ4n1-r2eQC.

[2] Fayçal Ait Aoudia and Jakob Hoydis. "End-to-end Waveform Learning Through Joint Optimization of Pulse and Constellation Shaping". In: *2021 IEEE Globecom Workshops (GC Wkshps)*. 2021, pp. 1–6. DOI: 10.1109/GCWkshps52748.2021.9681990.

[3] N. Benvenuto and M. Zorzi. *Principles of Communications Networks and Systems*. Wiley, 2011. ISBN: 9781119979821. URL: https://books.google.it/books?id=Y4MVojCoa1IC.

[4] Emil Bjornson and Pontus Giselsson. "Two Applications of Deep Learning in the Physical Layer of Communication Systems [Lecture Notes]". In: *IEEE Signal Processing Magazine* 37.5 (Sept. 2020), pp. 134–140. DOI: 10.1109/msp.2020.2996545. URL: https://doi.org/10.1109%2Fmsp.2020.2996545.

[5] Rick M. Bütler et al. "Model-Based Machine Learning for Joint Digital Backpropagation and PMD Compensation". In: *Journal of Lightwave Technology* 39.4 (2021), pp. 949–959. DOI: 10.1109/JLT.2020.3034047.

[6] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. "A Taxonomy for Semi-Supervised Learning Methods". In: *Semi-Supervised Learning*. 2006, pp. 15–31.

[7] David Cote. "Using machine learning in communication networks [Invited]". In: *Journal of Optical Communications and Networking* 10.10 (2018), pp. D100–D109. DOI: 10.1364/JOCN.10.00D100.

[8] C. Darken, J. Chang, and J. Moody. "Learning rate schedules for faster stochastic gradient search". In: *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*. 1992, pp. 3–12. DOI: 10.1109/NNSP.1992.253713.

[9] Sebastian Dörner et al. "Deep Learning Based Communication Over the Air". In: *IEEE Journal of Selected Topics in Signal Processing* 12.1 (2018), pp. 132–143. DOI: 10.1109/JSTSP.2017.2784180.

[10] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *J. Mach. Learn. Res.* 12.null (July 2011), pp. 2121–2159. ISSN: 1532-4435.

[11] Robert G. Gallager. *Principles of Digital Communication*. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511813498.

[12] Y. Gao et al. "112 Gb/s PAM-4 Using a Directly Modulated Laser with Linear Pre-Compensation and Nonlinear Post-Compensation". In: *ECOC 2016; 42nd European Conference on Optical Communication*. 2016, pp. 1–3.

[13] Rong Ge et al. "Escaping From Saddle Points - Online Stochastic Gradient for Tensor Decomposition". In: *CoRR* abs/1503.02101 (2015). arXiv: 1503.02101. URL: http://arxiv.org/abs/1503.02101.

[14] Ken Gentile. "The care and feeding of digital, pulse-shaping filters". In: 2002. URL: https://api.semanticscholar.org/CorpusID:62672765.

[15] Daniel George and E. A. Huerta. "Deep neural networks to enable real-time multimessenger astrophysics". In: *Physical Review D* 97.4 (Feb. 2018). DOI: 10.1103/physrevd.97.044039. URL: https://doi.org/10.1103%2Fphysrevd.97.044039.

[16] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[17] Christian Häger and Henry D. Pfister. "Physics-Based Deep Learning for Fiber-Optic Communication Systems". In: *IEEE Journal on Selected Areas in Communications* 39.1 (2021), pp. 280–294. DOI: 10.1109/JSAC.2020.3036950.

[18] M.H. Hayes. *Statistical Digital Signal Processing and Modeling*. Wiley India Pvt. Limited, 2009. ISBN: 9788126516100. URL: https://books.google.it/books?id=z0GqhOe9GNQC.

[19] S. Haykin and M. Moher. *An Introduction to Analog and Digital Communications*. Wiley, 2007. ISBN: 9780471432227. URL: https://books.google.it/books?id=0Q1LAQAAIAAJ.

[20] S.S. Haykin. *Adaptive Filter Theory*. Prentice-Hall information and system sciences series. Prentice Hall, 1996. ISBN: 9780133227604. URL: https://books.google.it/books?id=l78QAQAAMAAJ.

[21] S.S. Haykin. *Neural Networks and Learning Machines*. Pearson International Edition. Pearson, 2009. ISBN: 9780131293762. URL: https://books.google.it/books?id=KCwWOAAACAAJ.

[22] Zonglong He et al. "Experimental Demonstration of Learned Pulse Shaping Filter for Superchannels". In: *2022 Optical Fiber Communications Conference and Exhibition (OFC)*. 2022, pp. 1–3.

[23] Mohamed Ibnkahla. "Applications of neural networks to digital communications – a survey". In: *Signal Processing* 80.7 (2000), pp. 1185–1215. ISSN: 0165-1684. DOI: https://doi.org/10.1016/S0165-1684(00)00030-X. URL: https://www.sciencedirect.com/science/article/pii/S016516840000030X.

[24] Ankur Singh Kang and Vishal Sharma. "Pulse Shape Filtering in Wireless Communication-A Critical Analysis". In: *International Journal of Advanced Computer Science and Applications* 2 (2011). URL: https://api.semanticscholar.org/CorpusID:15429725.

[25] Boris Karanov et al. "End-to-End Deep Learning of Optical Fiber Communications". In: *Journal of Lightwave Technology* 36.20 (2018), pp. 4843–4855. DOI: 10.1109/JLT.2018.2865109.

[26] Boris Karanov et al. "End-to-end optimized transmission over dispersive intensity-modulated channels using bidirectional recurrent neural networks". In: *Opt. Express* 27.14 (July 2019), pp. 19650–19663. DOI: 10.1364/OE.27.019650. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-27-14-19650.

[27] Mansirat Kaur, Anand Kumar Shukla, and Saranjeet Kaur. "An Introduction to Machine Learning in a Nutshell". In: *2021 10th International Conference on System Modeling and Advancement in Research Trends (SMART)*. 2021, pp. 17–22. DOI: 10.1109/SMART52563.2021.9676315.

[28] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[29] Javier Mata et al. "Artificial intelligence (AI) methods in optical networks: A comprehensive survey". In: *Optical Switching and Networking* 28 (Apr. 2018), pp. 43–57. DOI: 10.1016/j.osn.2017.12.006. URL: https://doi.org/10.1016%2Fj.osn.2017.12.006.

[30] Rajesh Mehra and Ginne. "FPGA Based Gaussian Pulse Shaping Filter Using Distributed Arithmetic Algorithm". In: 2013. URL: https://api.semanticscholar.org/CorpusID:212591221.

[31] Tom M Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997.

[32] Tim O'Shea and Jakob Hoydis. "An Introduction to Machine Learning Communications Systems". In: *ArXiv* abs/1702.00832 (2017).

[33] Daniel Plabst et al. "Achievable Rates for Short-Reach Fiber-Optic Channels With Direct Detection". In: *Journal of Lightwave Technology* 40.12 (2022), pp. 3602–3613. DOI: 10.1109/JLT.2022.3149574.

[34] Daniel Plabst et al. "Wiener Filter for Short-Reach Fiber-Optic Links". In: *IEEE Communications Letters* 24.11 (2020), pp. 2546–2550. DOI: 10.1109/LCOMM.2020.3006921.

[35] Proakis. *Digital Communications 5th Edition*. McGraw Hill, 2007.

[36] John G. Proakis and Dimitris K Manolakis. *Digital Signal Processing (4th Edition)*. 4th ed. Prentice Hall, 2006. ISBN: 0131873741.

[37] Roi Rath et al. "Tomlinson–Harashima Precoding For Dispersion Uncompensated PAM-4 Transmission With Direct-Detection". In: *Journal of Lightwave Technology* 35.18 (2017), pp. 3909–3917. DOI: 10.1109/JLT.2017.2724032.

[38] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[39] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[40] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. URL: https://api.semanticscholar.org/CorpusID:205001834.

[41] David E. Rumelhart and James L. McClelland. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 1987, pp. 318–362.

[42] Meysam Sadeghi and Erik G. Larsson. "Adversarial Attacks on Deep-Learning Based Radio Signal Classification". In: *CoRR* abs/1808.07713 (2018). arXiv: 1808.07713. URL: http://arxiv.org/abs/1808.07713.

[43] Osvaldo Simeone. *A Brief Introduction to Machine Learning for Engineers*. 2018. arXiv: 1709.02840 [cs.LG].

[44] Osvaldo Simeone. "A Very Brief Introduction to Machine Learning With Applications to Communication Systems". In: *IEEE Transactions on Cognitive Communications and Networking* 4.4 (2018), pp. 648–664. DOI: 10.1109/TCCN.2018.2881442.

[45] Jinxiang Song et al. "End-to-end Autoencoder for Superchannel Transceivers with Hardware Impairment". In: *2021 Optical Fiber Communications Conference and Exhibition (OFC)*. 2021, pp. 1–3.

[46] J.C. Spall. "Implementation of the simultaneous perturbation algorithm for stochastic optimization". In: *IEEE Transactions on Aerospace and Electronic Systems* 34.3 (1998), pp. 817–823. DOI: 10.1109/7.705889.

[47] James C. Spall. "A one-measurement form of simultaneous perturbation stochastic approximation". In: *Automatica* 33.1 (1997), pp. 109–112. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(96)00149-5. URL: https://www.sciencedirect.com/science/article/pii/S0005109896001495.

[48] James C. Spall. "An overview of the simultaneous perturbation method for efficient optimization". In: *Johns Hopkins Apl Technical Digest* 19 (1998), pp. 482–492. URL: https://api.semanticscholar.org/CorpusID:7988308.

[49] Haoran Sun et al. "Learning to Optimize: Training Deep Neural Networks for Interference Management". In: *IEEE Transactions on Signal Processing* 66.20 (2018), pp. 5438–5453. DOI: 10.1109/TSP.2018.2866382.

[50] Shiliang Sun et al. "A Survey of Optimization Methods From a Machine Learning Perspective". In: *IEEE Transactions on Cybernetics* 50.8 (2020), pp. 3668–3681. DOI: 10.1109/TCYB.2019.2950779.

[51] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[52] S. Takeda and Y. Shigeoka. "An optical thin film Bessel filter for 40Gbit/sec -100GHz spacingD-WDM system". In: *2002 28TH European Conference on Optical Communication*. Vol. 5. 2002, pp. 1–2.

[53] Marc T. Thompson. "Chapter 14 - Analog Low-Pass Filters". In: *Intuitive Analog Circuit Design (Second Edition)*. Ed. by Marc T. Thompson. Second Edition. Boston: Newnes, 2014, pp. 531–583. ISBN: 978-0-12-405866-8. DOI: https://doi.org/10.1016/B978-0-12-405866-8.00014-0. URL: https://www.sciencedirect.com/science/article/pii/B9780124058668000140.

[54] Tim Uhlemann et al. "Deep-learning Autoencoder for Coherent and Nonlinear Optical Communication". In: *Photonic Networks; 21th ITG-Symposium*. 2020, pp. 1–8.

[55] Tim Uhlemann et al. "Introducing γ-lifting for Learning Nonlinear Pulse Shaping in Coherent Optical Communication". In: *Photonic Networks; 23th ITG-Symposium*. 2022, pp. 1–8.

[56] *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014, pp. I–XVI, 1–397.

[57] Jinlong Wei et al. "400 Gigabit Ethernet using advanced modulation formats: Performance, complexity, and power dissipation". In: *IEEE Communications Magazine* 53.2 (2015), pp. 182–189. DOI: 10.1109/MCOM.2015.7045407.

[58] Mark Wickert and David Peckham. "Matched Filter Mismatch Losses in MPSK and MQAM Using Semi-Analytic BEP Modeling". In: Jan. 2020, pp. 98–106. DOI: 10.25080/Majora-342d178e-00e.

[59] Xiang-gen Xia. "A family of pulse-shaping filters with ISI-free matched and unmatched filter properties". In: *IEEE Transactions on Communications* 45.10 (1997), pp. 1157–1158. DOI: 10.1109/26.634674.

[60] Yapeng Xie et al. "Machine Learning Applications for Short Reach Optical Communication". In: *Photonics* 9.1 (2022). ISSN: 2304-6732. DOI: 10.3390/photonics9010030. URL: https://www.mdpi.com/2304-6732/9/1/30.

[61] Qiang Zhang et al. “Transmission of single lane 128 Gbit/s PAM-4 signals over an 80 km SSMF link, enabled by DDMZM aided dispersion precompensation”. In: *Opt. Express* 24.21 (Oct. 2016), pp. 24580–24591. DOI: 10.1364/OE.24.024580. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-24-21-24580.

[62] Kangping Zhong et al. “Digital Signal Processing for Short-Reach Optical Communications: A Review of Current Technologies and Future Trends”. In: *Journal of Lightwave Technology* 36.2 (2018), pp. 377–400. DOI: 10.1109/JLT.2018.2793881.

[63] Honghang Zhou et al. “Recent Advances in Equalization Technologies for Short-Reach Optical Links Based on PAM4 Modulation: A Review”. In: *Applied Sciences* 9.11 (June 2019), p. 2342. ISSN: 2076-3417. DOI: 10.3390/app9112342. URL: http://dx.doi.org/10.3390/app9112342.

[64] Darko Zibar et al. “Machine Learning Techniques in Optical Communication”. In: *Journal of Lightwave Technology* 34.6 (2016), pp. 1442–1452. DOI: 10.1109/JLT.2015.2508502.

Acknowledgments

I would like to express my sincere gratitude to my Supervisors, Professor Francesco Da Ros and Professor Nicola Laurenti.
Professor Da Ros, my Supervisor in Denmark, gave me the opportunity to join his Research Group at the Technical University of Denmark. With his great generosity, teachings and advice, he encouraged me to expand my knowledge and to extend my competences.
I am deeply grateful to Professor Laurenti, my Supervisor in Italy, for enabling this collaboration and for giving substance to this project, thanks to his advice, valuable contributions and his encouragement.
I am also thankful to Professor Darko Zibar, who welcomed me in his Research Group and gave me full trust.
I would like to give special thanks to Dr. Hernandez Fernandez because he was always available to help me in my daily work.
Last but not least, I had the honour to collaborate with the Machine Learning in Photonic Systems research group that warmly welcomed me and considered me a member of the group since day one. Forever grateful.