# Neural Networks for Optical Channel Equalization in High Speed Communication Systems

**Mémoire**

**Sai Chandra Kumari Kalla**

**Maîtrise en génie électrique - avec mémoire**
Maître ès sciences (M. Sc.)

Québec, Canada

# Neural Networks for Optical Channel Equalization in High Speed Communication Systems

**Mémoire**

**Sai Chandra Kumari Kalla**

Sous la direction de:

Leslie Ann Rusch, directrice de recherche

# Résumé

La demande future de bande passante pour les données dépassera les capacités des systèmes de communication optique actuels, qui approchent de leurs limites en raison des limitations de la bande passante électrique des composants de l'émetteur. L'interférence intersymbole (ISI) due à cette limitation de bande est le principal facteur de dégradation pour atteindre des débits de données élevés. Dans ce mémoire, nous étudions plusieurs techniques de réseaux neuronaux (NN) pour combattre les limites physiques des composants de l'émetteur pilotés à des débits de données élevés et exploitant les formats de modulation avancés avec une détection cohérente.

Notre objectif principal avec les NN comme égaliseurs de canaux ISI est de surmonter les limites des récepteurs optimaux conventionnels, en fournissant une complexité évolutive moindre et une solution quasi optimale. Nous proposons une nouvelle architecture bidirectionnelle profonde de mémoire à long terme (BiLSTM), qui est efficace pour atténuer les graves problèmes d'ISI causés par les composants à bande limitée. Pour la première fois, nous démontrons par simulation que notre BiLSTM profonde proposée atteint le même taux d'erreur sur les bits (TEB) qu'un estimateur de séquence à maximum de vraisemblance (MLSE) optimal pour la modulation MDPQ.

Les NN étant des modèles pilotés par les données, leurs performances dépendent fortement de la qualité des données d'entrée. Nous démontrons comment les performances du BiLSTM profond réalisable se dégradent avec l'augmentation de l'ordre de modulation. Nous examinons également l'impact de la sévérité de l'ISI et de la longueur de la mémoire du canal sur les performances de la BiLSTM profonde. Nous étudions les performances de divers canaux synthétiques à bande limitée ainsi qu'un canal optique mesuré à 100 Gbaud en utilisant un modulateur photonique au silicium (SiP) de 35 GHz. La gravité ISI de ces canaux est quantifiée grâce à une nouvelle vue graphique des performances basée sur les écarts de performance de base entre les solutions optimales linéaires et non linéaires classiques. Aux ordres QAM supérieurs à la QPSK, nous quantifions l'écart de performance BiLSTM profond par rapport à la MLSE optimale à mesure que la sévérité ISI augmente. Alors qu'elle s'approche des performances optimales de la MLSE à 8QAM et 16QAM avec une pénalité, elle est capable de dépasser largement la solution optimale linéaire à 32QAM. Plus important encore, l'avantage de l'utilisation de modèles d'auto-apprentissage comme les NN est leur capacité à apprendre

le canal pendant la formation, alors que la MLSE optimale nécessite des informations précises sur l'état du canal.

# Abstract

The future demand for the data bandwidth will surpass the capabilities of current optical communication systems, which are approaching their limits due to the electrical bandwidth limitations of the transmitter components. Inter-symbol interference (ISI) due to this band-limitation is the major degradation factor to achieve high data rates. In this thesis, we investigate several neural network (NN) techniques to combat the physical limits of the transmitter components driven at high data rates and exploiting the advanced modulation formats with coherent detection.

Our main focus with NNs as ISI channel equalizers is to overcome the limitations of conventional optimal receivers, by providing lower scalable complexity and near optimal solution. We propose a novel deep bidirectional long short-term memory (BiLSTM) architecture, that is effective in mitigating severe ISI caused by bandlimited components. For the first time, we demonstrate via simulation that our proposed deep BiLSTM achieves the same bit error rate (BER) performance as an optimal maximum likelihood sequence estimator (MLSE) for QPSK modulation.

The NNs being data-driven models, their performance acutely depends on input data quality. We demonstrate how the achievable deep BiLSTM performance degrades with the increase in modulation order. We also examine the impact of ISI severity and channel memory length on deep BiLSTM performance.We investigate the performances of various synthetic band-limited channels along with a measured optical channel at 100 Gbaud using a 35 GHz silicon photonic (SiP) modulator. The ISI severity of these channels is quantified with a new graphical view of performance based on the baseline performance gaps between conventional linear and nonlinear optimal solutions. At QAM orders above QPSK, we quantify deep BiLSTM performance deviation from the optimal MLSE as ISI severity increases. While deep BiLSTM approaches the optimal MLSE performance at 8QAM and 16QAM with a penalty, it is able to greatly surpass the linear optimal solution at 32QAM. More importantly, the advantage of using self learning models like NNs is their ability to learn the channel during the training, while the optimal MLSE requires accurate channel state information.

# Table of Contents

# List of Figures

# List of abbreviations

| | |
|---|---|
| ANN | artificial neural network |
| AWGN | additive white Gaussian noise |
| BER | bit error rate |
| BiLSTM | bidirectional long-short term memory |
| BP | back propagation |
| BW | band-width |
| CE | cross-entropy |
| CSI | channel state information |
| CNN | convolution neural network |
| DAC | digital-to-analog convertor |
| DFB | decision feedback |
| DL | deep learning |
| DNN | deep neural network |
| DSP | digital signal processing |
| FEC | forward error correction |
| FFNN | feed-forward neural network |
| GPU | graphical processing unit |
| ISI | inter-symbol interference |
| LFFNN | linear feed-forward neural network |
| LMLP | linear multilayer perceptron |
| LSTM | long-short term memory |

| | |
|---|---|
| ML | machine learning |
| MLSE | maximum likelihood sequence estimation |
| MP | multipath |
| MMSE | minimum mean squared error |
| MSE | mean square error |
| MZM | Mach-Zehnder modulator |
| NLP | natural language processing |
| NN | neural network |
| OFDM | orthogonal frequency division multiplexing |
| PAM | pulse-amplitude modulation |
| PSD | power spectral density |
| QAM | quadrature amplitude modulation |
| QPSK | quadrature phase shift keying |
| RNN | recurrent neural network |
| SGD | stochastic gradient descent |
| SG | super-Gaussian |
| SiP | silicon photonic |
| SNR | signal-to-noise ratio |
| TL | transfer learning |
| ZF | zero-forcing |

*For my family, who always believed in me and stood by my side. I am forever grateful for their continuous and unparalleled love, support, and encouragement.*

If you fail, never give up because F.A.I.L. means "First Attempt In Learning"

---

APJ Abdul Kalam

# Acknowledgements

I take this opportunity to express my immense gratitude to all the people who helped me along the way in completing my project. It is my pleasure that I am penning down these lines.

Firstly, I am very grateful to Prof. Leslie Ann Rusch, my supervisor, mentor, and my inspiration. I want to thank her for welcoming me as an intern at COPL and giving me the desire to continue for my Masters. She always helped me even with my basic questions and taught me a lot about research and life, that helped me to sharpen my thinking, research and presentation skills. Her expertise was invaluable in formulating this research problem and methodologies; her motivating words, friendly nature, valuable guidance and constructive criticism shaped this research by overcoming the challenges. I feel blessed to have worked with her twice.

Next I would like to thank Prof. Christian Gagné for his continuous support, even though I was not his student. His diverse input and valuable suggestions about the machine learning helped me to keep an open mind to the research challenges we faced.

Further, I want to also thank to Prof. Ming Zeng, Jiachuan Lin, and Sasan for their valuable comments and feedback all the way during this project. Thanks to Sami, Eloise, and Hugo who helped me with my machine learning course for understanding Python and dealing with French. A big thanks to Sumanth, for helping me to better understand the LSTMs and its challenges. Thanks to REPOL and BVE for organizing amazing activities that made me appreciate my time during my Masters. Moreover, I am grateful to the entire team of COPL and LCO for providing me with the best work culture, where everyone can share their ideas and help each other. Special thanks to Amir, Mai, Charles, Sammaneh, Xuan, Lipeng, Zibo, and Satyendra for all those memorable moments and your moral support.

Many thanks to my friends in Quebec and back in India for constantly reaching out and making me feel comfortable in a place away from home. Finally and most important thanks to my wonderful family for their unconditional love and support; and for always being with me through my thick and thin, and being the reason I push myself harder every day.

Doing research in a foreign country, I faced a lot of challenges and fears; but I grew up both personally and professionally during this journey. To everyone who constantly encouraged and believed in me to reach this stage of my life, this work is all yours.

# Foreword

Chapter three of this thesis is based on the journal paper [2], which is submitted to JLT. Some parts of this work are both accepted and published in the conference proceedings at IPC 2020 [3] and CLEO 2020 [4]. I was the main contributor to these papers. All the submissions and publications are based on the simulations using Python and MATLAB platforms. In what follows, the responsibility and contribution of each co-author of the papers used for this thesis will be explained.

- Sai Chandra Kumari Kalla, Christian Gagné, and Leslie Ann Rusch, "Recurrent Neural Networks Achieving MLSE Performance for Optical Channel Equalization," submitted, *Journal of Lightwave Technology (JLT)*, October 2020. This paper reports the record breaking results via simulations, that for QPSK transmission the proposed deep bidirectional long short term (BiLSTM) architecture achieves the optimal bit error rates of maximum likelihood sequence estimator (MLSE). This work also illustrates the significance of data quality on achievable BiLSTM performance.

  I developed the deep BiLSTM architecture and performed all the simulations presented in this paper. Leslie Rusch provided overall guidance and supervised this research. Christian Gagné, professor at Université Laval, provided many valuable suggestions and comments on overcoming the challenges in the neural networks (NN). The paper was prepared by me and revised by the co-authors.

- Sai Chandra Kumari Kalla, Rizan Homayoun Nejad, Sasan Zhalehpour, and Leslie A. Rusch, "Neural Nets to Approach Optimal Receivers for High Speed Optical Communication," *In CLEO: Science and Innovations*, pp. STh4M-4, Optical Society of America, May 2020. This paper shows via simulation that for a QPSK transmission, two nonlinear NN solutions approach the performance of maximum likelihood sequence estimator (MLSE) for experimental frequency responses of a silicon photonic (SiP) modulator and coherent detection.

I developed the two nonlinear NNs and a baseline linear NN that was used in this paper. Leslie Rusch provided overall guidance and supervised this research. Sasan Zhalehpour, a PhD student under Leslie Rusch, provided the measured channel taps for experimental 100 Gbaud optical SiP channel setup in [5]. Rizan Homayoun Nejad, a PhD student with Leslie Rusch, contributed by estimating the BER values for the conventional optimal receivers. The paper was written by me and revised by the co-authors.

- Sai Chandra Kumari Kalla and Leslie Ann Rusch, "Recurrent neural nets achieving MLSE performance in bandlimited optical channels," in *2020 IEEE Photonics Conference (IPC)*, pp. MA3-3, September 2020. This paper demonstrates our proposed BiLSTM architecture achieves the same BER values as that of MLSE receiver for two bandlimited channels (experimental and synthetic), for QPSK transmission.

  I developed the BiLSTM architecture that was used in this paper. Leslie Rusch provided overall guidance and supervised this research. The paper was written by me and revised by Leslie Rusch.

# Introduction

## I.1 Motivation

Today, we live in what many call the Information Age [6], in which people can access information and knowledge with the touch of a button. Data has become an important part of everyday lives, and we rely on data services providing digital information and connecting us via email, video calls, and social media. Consumer demand for data bandwidth is growing exponentially, driven by bandwidth-hungry applications (from high quality video services to low-latency real-time communications) on ever increasing number of devices. Escalating data concerns are rampant with the meta data applications such as, cloud applications, machine learning markets, and internet of things (IOT) [7].

Concurrently, fiber optics [8] revolutionized the communication systems (since 1970s) and played a major role to satisfy the ever growing bandwidth hunger through the low loss and bandwidth provided by the optical fibres [9]. In the past decades, the net transmitting data rates of optical communications systems have been enhanced by multiple technological breakthroughs that include coherent detection, advanced modulation formats, and digital signal processing (DSP) [10; 11; 12]. Available data rates have reached terabytes and petabytes.

According to Cisco visual networking index [13], global IP traffic will reach 278.0 exabyte (one exabyte is equivalent to one million terabytes) per month by 2021, up from 96.0 exabyte per month in 2016. To achieve such high data rates in optical communication systems, aggressive bit rates are targeted using higher order quadrature amplitude modulation (QAM) formats and coherent detection. However, targeting higher data rates in optical systems can lead to transmission impairments which limit the overall system performance. The electrical bandwidth of the transceiver components such as digital-to-analog converter (DAC), Mach-Zehnder-modulator (MZM) are limited, and pose the major challenges in achieving higher data rates. Despite the use of spectrally efficient higher order QAM, the required data rates are greater than the available channel bandwidth. When high baud rate signals are transmitted through these band-limited components, the high frequencies are attenuated, leading to pulse spreading in the time domain. The inter-symbol-interference (ISI) caused due to the pulse spreading is the principal impairment in high speed optical communication systems with

band-limited components. Moreover, with the use of higher QAM, signals are less robust to noise and ISI distortion.

To achieve reliable transmission at high baud rates, one can use post-compensation techniques to mitigate ISI. In past few years, many channel equalisation techniques have been proposed, both optimal and sub optimal solutions. Minimum mean squared error (MMSE) equalizer [14; 15] combats ISI and is the optimal linear, one-shot receiver for Gaussian noise channels. However, MMSE performance quickly deteriorates with band limitation and QAM order; it only provides a sub optimal solution for severe ISI mitigation. The maximum likelihood sequence estimator (MLSE) [14; 16] provides the optimal nonlinear solution by examining all possible sequences to select the most probable transmitted sequence. However, the MLSE has computational complexity that increases exponentially with the modulation order and the ISI memory length, making it's implementation impossible at high modulation order. Moreover, its achievable performance relies on the quality of available channel state information (CSI). The complexity of underlying channels is increasing, and thus mathematical channel modelling is very challenging. This can lead to CSI estimation error, which deviates the MLSE performance away from the optimal performance.

In recent years, machine learning [17] and deep learning [18; 19] techniques have lead to record breaking results in many areas of communications [20; 21]. These techniques promise to resolve emerging problems within these research directions. Especially, neural networks (NN) have shown remarkable success in the fields of signal processing [22; 23]. The main advantage of these techniques is their capability to learn the function from the training data, without the need of being explicitly programmed. They also hold the potential to improve reliability, generality and latency (complexity) of the model.

In this work, we explore different NN architectures for mitigating severe ISI caused by band-limited components, thereby targeting the reliable high speed optical communications. Our examinations span a simple feedforward NN (FFNN) to a more complex recurrent NN, i.e., long-short term memory (LSTM) architecture [24]. We compare the performance of our NNs vis-à-vis MMSE and MLSE solutions for both synthetic and experimental channels, at different modulation orders. We demonstrate that a LSTM architecture with emphasis on sequential decisions can effectively mitigate severe ISI and achieve the optimal MLSE performance. The role of training data quality on attainable NN performance is also illustrated.

## I.2 Thesis Outline

The rest of the thesis is organized as follows:

In Chapter 1, we review the fundamental building blocks of the coherent optical communication systems with the focus on band-limited components. We simulate a typical band-limited

channel and describe the channel distortions for a transmitted signal at low and high baud rates. We then introduce our experimentally measured 100 Gbaud optical SiP modulator with a 3dB bandwidth of 35 GHz. We describe impact of ISI and show the BER deterioration at high baud rate transmissions. Later, we introduce the conventional linear (MMSE) and nonlinear (MLSE) optimal equalization techniques, describe their implementation, and also their limitations for targeting high data rates.

In Chapter 2, we first present the fundamental and vital concepts of NN training, with the focus on solutions employed in Chapter 3. We describe the basics of NN architecture and the error criteria used for training our NN equalizers. The training of the NN is then detailed with the emphasis on optimization and learning algorithms. Later, we provide an extensive literature review of NNs for ISI channel equalization and then train our first baseline NN (linear feedforward NN) for ISI mitigation. We show it can only achieve MMSE performance. We highlight that deep learning architectures are the key to identify hidden sequential patterns in the data and thereby target optimal MLSE performance.

In Chapter 3, we investigate both feedforward and recurrent NN architectures for severe ISI mitigation. For a robust QPSK transmission, we exploit different characteristics of MLSE (non-linearity, sequence detection) to target the optimal performance; propose a novel deep BiLSTM architecture that strongly emphasizes on sequential correlations. For the first time, we successfully demonstrated that deep BiLSTM achieves optimal MLSE performance for QPSK. To generalize our conclusions, in addition to the experimental SiP channel we also examine the performance for two families of synthetic ISI channels (one multipath family and another super Gaussian family) to sweep through ISI severity. We demonstrate how the deep BiLSTM performance deviates from the optimal solution with the increase in modulation order and ISI severity, but always achieves significant gain over MMSE. Finally, we provide an insight into the scalability of MLSE and deep BiLSTM architectures.

In the conclusion, we summarize the important contributions of this work and highlight the future scope and impacts. Finally, in an appendix we present the framework for our simulations along with pseudo code for training deep BiLSTM.

# Chapter 1

# Conventional Equalization in Coherent Communication Systems

In this chapter, we introduce the basic coherent communication system and the need for channel equalization. We also describe the conventional optimal receivers and their performance. Section 1.1 introduces the principal system impairments in coherent communication systems. Section 1.2 focuses on the memory effects in the high speed communication systems, with bandlimited components. In Section 1.3, we introduce the conventional equalization techniques to mitigate ISI and describe their performance in Section 1.4.

## 1.1 Coherent Communication

### 1.1.1 Coherent detection

Coherent communication [10] is a promising approach for spectral efficient high-speed optical communication. The input data (information to be transmitted) is generally modeled as a random sequence of binary bits. In coherent systems, the information (input) bits modulate both the amplitude and phase of the carrier wave, known as quadrature amplitude modulation (QAM) [11]. In QAM, $\log_2 M$ bits are encoded into $M$ symbols with signal coordinates (I, Q) where $M$ is the QAM modulation order, and I and Q are in-phase and quadrature components of the signal. In Fig. 1.1, we show several M-QAM symbol constellations plotted in the I, Q plane. As the constellation size ($M$) increases, more bits can be encoded into a symbol, increasing the spectral efficiency. In a constrained average power transmission, the distance between the symbols decreases with the increase in modulation order, leading to lower power efficiency.

The functional block diagram shown in Fig. 1.2 illustrates the signal flow through a typical communication system. The modulated QAM signal is generated at the transmitter and transmitted through a channel which distorts the signal. At the receiver, additive white

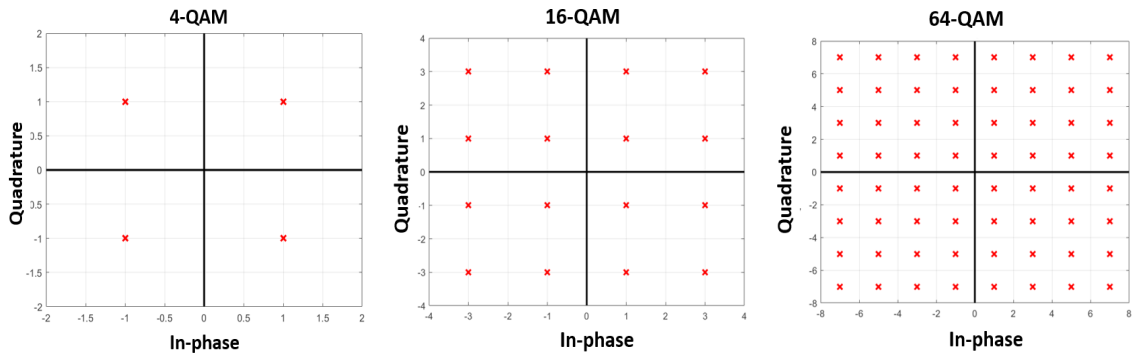



Figure 1.1 – Constellation diagrams.

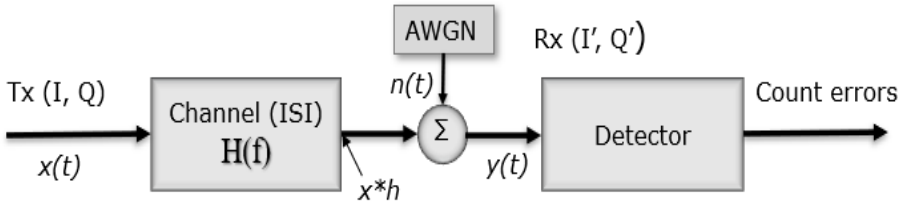



Figure 1.2 – Block diagram of typical communication system.

Gaussian noise (AWGN) corrupts the received symbols. The detector maps the received signals to the transmitted $M$ symbols using decision boundaries. For equiprobable symbols, these decision boundaries map the received signal to the closest symbol in the constellation.

The channel distortion effects can be linear, nonlinear or both. In our work, we focus on inter symbol interference (ISI), a linear distortion introduced by a bandlimited channel. This methodology could be extended in the future to other scenarios to assess its effectiveness more generally.

In the absence of noise and distortion effects the received signals would exactly coincide with the points in Fig. 1.1. In practical systems, we instead observe a cloud of received symbols around the transmitted symbols. This scattering is due to displacement by noise and distortion of the signal both in amplitude and phase. Thus during detection, symbol errors occur and consequently bit errors occur, which decreases the overall system performance. In the next sections, we describe these impairments.

### 1.1.2 Additive white Gaussian noise (AWGN)

In coherent systems, thermal noise is introduced at the receivers during detection. This noise is well modeled as additive random noise with a Gaussian distribution. The auto-correlation function is a Dirac delta function, and the power spectral density (PSD) is flat. Due to the flat PSD, it is commonly referred as additive white Gaussian noise (AWGN). Noise samples

are independent and identically distributed due to the white nature of the noise. The signal-to-noise-ratio (SNR) is defined as the ratio of the input signal power to the noise power. For a two-sided PSD of $N/2$, the noise samples will be Gaussian with zero mean and variance $\sigma^2 = \frac{N}{2}$. Thus with the increase in noise power, noise variance increases as seen in the increased cloud around each symbol. For a fixed transmit power, increasing the modulation order leads to closer-packed symbols and thus less noise tolerance.

### 1.1.3 Inter-symbol interference (ISI)

The ISI is a linear distortion introduced by a bandlimited channel. That is, when the channel bandwidth is significantly smaller than the signal bandwidth, pulse spreading occurs. The spread pulses interfere with one another, as shown in Fig 1.3. The symbol being sampled is thus influenced by past and future symbols. This unwanted phenomenon leads to a cloud, much like that due to AWGN. As with AWGN, ISI impact also increases with the modulation order. However, while AWGN is completely unpredictable, ISI has structure and can be predicted. Most importantly, we observe a BER performance floor with ISI, i.e., BER performance is saturated. Increasing signal power (SNR) does not decrease ISI, though it does lessen the impact of AWGN.

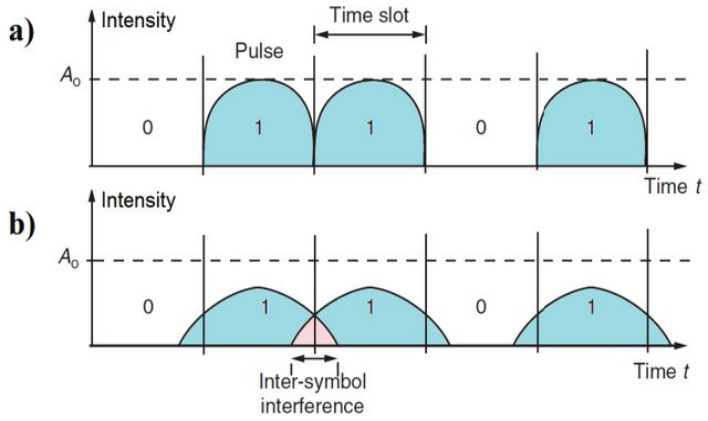



Figure 1.3 – Inter-symbol interference in time domain [1].

The ISI is also observed in dispersive channels (e.g., due to chromatic dispersion in optical fibers), in multi-path fading channels (e.g., wireless channels) and in any system with bandlimited devices. In this thesis, we focus on ISI introduced by bandlimited channels when targeting high data rates.

## 1.2 High Speed Communication in Coherent Systems

High speed optical communication systems tend to be highly bandlimited. Usually data rate targets are either comparable or higher than the channel bandwidth. Transmitting higher data rates through these bandlimited channels introduces distortion and increases BER.

In this section, firstly we model a bandwidth-limited channel and describe the ISI due to high data rate targets through these channels. Then, we introduce our examined bandwidth-limited optical channel, measured at 100 Gbaud using a silicon photonic (SiP) modulator [5]. Finally, we illustrate its BER performance for different data rates.

### 1.2.1 Simulating performance of a bandwidth-limited channel

A bandwidth-limited channel can be modelled either in the frequency domain as a frequency response or in the time domain as an impulse response. In Fig. 1.2, suppose the channel frequency response is $H(f)$ and the PSD of the input signal is $X(f)$, then the PSD of the output is $X(f)H(f)$. The channel can also be equivalently modeled in the time domain as a linear convolution filter [14]. If the channel impulse response is $h(t)$ and the transmitted input signal is $x(t)$, the output of the bandwidth-limited channel is given by $x(t) \circledast h(t)$, where $\circledast$ denotes the convolution operation.

Noise, $n(t)$, at the receiver corrupts the channel output and the detector observes a noisy version of the received signal. The final output, $y(t)$, is given by

$$y(t) = x(t) \circledast h(t) + n(t) \tag{1.1}$$

Suppose the receiver samples the signal with sampling time $T$, the sampled output can be expressed as

$$y(mT) = y(t)|_{t=mT} = (x(t) \circledast h(t))|_{t=mT} + n(mT) \tag{1.2}$$

where noise samples are Gaussian distributed with zero mean, $\sigma^2$ variance, i.e., $n(T) \sim N(0, \sigma^2)$. We assume the noise is limited to the bandwidth of the receiver, i.e., the signal bandwidth. The impact of the noise is determined by SNR. In simulations, we model a bandwidth-limited channels by discrete $h(t)$ with $n$ taps.

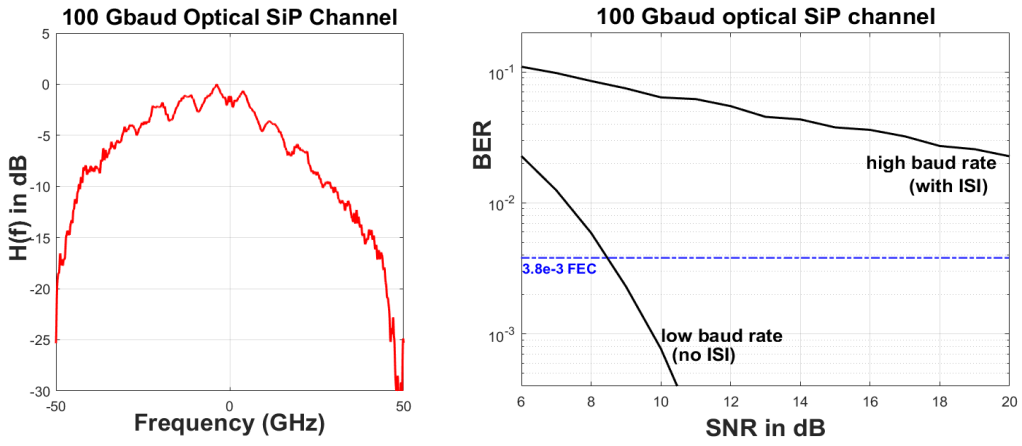
The system performance is evaluated via estimating BER. In our simulations, we estimate the BER vs SNR performance using Monte Carlo simulations. Forward error correction (FEC) [25; 26] can be used to improve performance. The FEC coding (or channel coding) adds overhead to the information bits at the transmitter. At the receiver, these code words are decoded and the errors are corrected. Using FEC codes, a big improvement in BER performance can be achieved. However, to achieve error free transmission (BER $\approx 10^{-15}$), the BER of the un-coded channel must be less than a threshold. The common FEC codes used in the optical communications industry include 7% and 20% FEC overhead; these thresholds are measured at the BER values of 3.8e-3 and 2.4e-2, respectively [27]. For an un-coded channel, we focus on achieving 7% FEC threshold, i.e., 3.8e-3 BER.

### 1.2.2 ISI from high speed transmission

Transmitting high data rates can require greater bandwidth than that available for the channel. Using higher order QAM modulation can increase spectral efficiency, however, pushing to higher baud rates will eventually exceed the available channel bandwidth. This results in severe attenuation at high frequencies i.e., signal distortion occurs. A bandwidth-limited channel increases the signal rise and fall times, spreading the pulses, and leading to ISI. We say the channel introduces memory effects, i.e., symbols are influenced by a surrounding pattern of symbols.

Suppose the signal bandwidth is $BW_s$ and channel bandwidth is $BW_{ch}$. There is no channel distortion (and no ISI) if $BW_s < BW_{ch}$, i.e., for low data rate transmission. For high data rate transmission, $BW_s \gg BW_{ch}$, the signal is distorted. Nyquist pulse shaping [14; 28] is usually employed at the transmitter side to mitigate this effect by reducing signal bandwidth as much as possible. Despite Nyquist shaping, ISI remains for very high data rate targets.

Our work is motivated by the challenges in silicon-photonic (SiP) modulator operation at baud rates (symbol rates) that greatly exceed their nominal channel bandwidth. From the experimental setup in [5], we estimate the SiP modulator frequency response with 35 GHz 3 dB bandwidth. The measured 513-tap channel frequency response is given in Fig.1.4(a). In our simulations, the 513-tap estimated channel is approximated by 3 taps ([0.25+0.38i ; 0.75+0.84i ; 0.42+0.09i]) for ease of comparison with other channels examined in Chapter 3. The coefficients are complex due to non-ideal behavior of the modulator.




(a) (b)

Figure 1.4 – (a) Frequency response and (b) BER performance for 100 Gbaud Optical SiP channel.

For baud rates below 10 Gbaud, the SiP channel acts as an all-pass filter and the signal is not distorted. For higher baud rate targets such as 100 Gbaud, the optical SiP channel is bandlimited and the output is distorted and corrupted by ISI. Increasing the signal baud rate,

increases the ISI impact.

In Fig. 1.4 (b) we plot the BER vs. SNR performance for 100 Gbaud optical SiP channel at low and high baud rate operations, for QPSK transmission. In the absence of ISI, i.e., low baud rate transmission, the BER is given by theoretical calculations for an AWGN channel. For QPSK transmission, this equation is approximately given as below [29]:

$$P_b = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{1.3}$$

where $\frac{E_b}{N_0}$ is the digital SNR (or SNR per bit), and erfc$(\cdot)$ is the complementary error function traditionally defined as

$$\text{erfc}(x) = \frac{2}{\pi}\int_x^\infty e^{-u^2}du \tag{1.4}$$

For such a system, the matched filter detector provides the optimal BER performance; this is plotted in the solid no-ISI curve in Fig. 1.4(b).

In the presence of ISI, i.e., for high baud rate transmissions, the BER can be estimated via Monte Carlo techniques. The received symbols are decided based on the nearest constellation point (a QAM demodulator) and the BER is estimated; this is given by the solid "with-ISI" curve in Fig. 1.4(b). We observe the onset of a BER floor, as is typical for ISI. If the BER floor is higher than the FEC threshold, error correction will be ineffective. Therefore, high data rate information cannot be reliably transmitted over bandlimited channels using a conventional matched filter. Channel equalizers can be used to compensate the ISI at the receiver side before detection, as described in following section.

## 1.3 Conventional Equalization

Post-compensation or equalization techniques aim to remove the ISI from the received symbols. In our work, we consider both linear and nonlinear equalization techniques. The linear equalizers provide a low-complexity solution, but their performance is limited. Whereas the nonlinear equalizers perform better, but at the cost of higher complexity. In this section, we first introduce the conventional equalizers for ISI mitigation, with the focus on optimal receivers. Later, we illustrate their implementation and performance.

### 1.3.1 Linear equalizers

The zero-forcing (ZF) equalizer and the minimum mean squared error (MMSE) equalizer are popular linear equalizers for mitigating ISI [14; 15]. The ZF equalizer employs an infinite tap filter and adapts its weights to remove ISI by finding the channel inverse. However, it does not consider the noise factor in its design, and thus the noise is enhanced while removing ISI.

The ZF equalizers cannot provide the best BER performance in the practical channels with noise; it worsens the input BER for channels with spectral nulls or severe attenuation. In other words, the ZF equalizer cannot provide optimal BER performance for mitigating severe ISI in bandlimited channels.

The MMSE equalizer provides the linear optimal solution for mitigating ISI. In contrast to the ZF equalizer, the MMSE equalizer considers the effects of both ISI and AWGN. This equalizer can be either model based or data driven. For a known channel, that is for perfect channel state information (CSI), the exact MMSE equalizer can be found. Alternately, when CSI is not available, in training-based equalization, we adapt its tap weight coefficients to minimise the mean square value of the error. The error is

$$\epsilon_k = z_k - s_k \tag{1.5}$$

where $s_k$ is the transmitted symbol at $k$th signaling interval and $z_k$ is the estimate of that symbol at the output of the equalizer, defined as

$$z_k = \sum_{j=-K}^{j=K} c_j y_{k-j} \tag{1.6}$$

where $y_k$ is the received symbol at $k$th signaling interval and $c_j$ are the $2K+1$ tap weight coefficients of the filter. The performance index for the MSE criterion, denoted by $J$, is defined as

$$J = E|\epsilon_k|^2 = E|s_k - z_k|^2 \tag{1.7}$$

where $E(\cdot)$ is the expected value function.

The MMSE equalizer is relatively easy to realize and implement (see section 1.3.3). However, there are a few drawbacks of this linear optimal equalizer. Even though it provides an efficient solution for low ISI channels, the MMSE performance depends on channel frequency response; its performance quickly deteriorates with the increased band-limitation (or ISI impact) in the channels. Thus, the MMSE solution is insufficient for reliably transmitting high data rates signals in bandlimited channels, as it could only provide a sub-optimal solution for compensating severe ISI.

### 1.3.2 Nonlinear equalizers

The linear equalizers suffer more from the noise enhancement and the channel attenuation, than the non-linear equalizers. The decision-feedback (DFB) equalizer and maximum likelihood sequence estimator (MLSE) equalizer are useful promising nonlinear solutions to improve the BER performance in severely bandlimited channels.

The DFB equalizer does symbol-by-symbol detection and employs two filters, i.e., a feedback and a feed-forward filter. Using both these filters, it considers the decisions of previous equalized symbols and removes the ISI from the present symbol. The performance of the DFB

equalizer is much better than that of MMSE equalizer [14]. However in severely bandlimited channels, it's performance quickly deteriorates, as the probability of incorrect previous decisions increases.

In contrast to all the previous equalizers mentioned, the MLSE equalizer [14; 16] provides an excellent solution for mitigating ISI; it is the optimal nonlinear receiver.

Due to memory effects, the information in the current symbol influences surrounding symbols. It is intuitive that the optimal receiver (for mitigating this effect) should not only observe a part of the received symbols to decide on current symbol, but instead the whole received sequence. The MLSE provides the nonlinear optimal performance by employing sequence detection instead of symbol-by symbol (or one-shot) detection. The channel output with the memory effect can be conveniently represented by a trellis structure [16]. Using this trellis, the MLSE equalizer outputs the minimum error solution by finding the most probable trellis path, which is equivalent to selecting the most probable transmitted sequence.

Despite it being a nonlinear optimal equalizer, the MLSE equalizer has a few prominent drawbacks to achieve this optimal performance. The MLSE is a model-based equalizer and requires prior and perfect CSI to provide the optimal solution. Underlying channel complexity can be high, thus estimating perfect CSI is not always possible. The error in the CSI estimation could severely degrade performance of this model-based receiver, and deviate from the optimal performance. Furthermore, the MLSE equalizer has complexity that grows exponentially with modulation order and memory length. Suppose the channel is represented by $L$ taps in an FIR filter, i.e., channel memory of $L-1$ symbols, and the modulation order is $M$. The MLSE computes $M^{L-1}$ state metrics for each received symbol to keep the most probable path. While targeting very high data rates, higher order ($M$) QAM is usually implemented and with the severe band-limitation, memory length ($L$) also increases. Thus the solution provided by MLSE becomes infeasible to implement.

### 1.3.3 Implementation of optimal equalizers in MATLAB

In this section, we describe the implementation of the linear-optimal MMSE and nonlinear-optimal MLSE equalizers in MATLAB.

We implement the training based MMSE equalizer using Monte Carlo simulations in MATLAB. The inputs to this equalizer are: transmitted symbols ($s_k$), received symbols ($y_k$) and number of filter taps ($2K+1$). The MMSE tap weight coefficients ($c_k$) that minimizes the mean squared error in Eq. 1.7 are yielded from the following set of simultaneous equations [14]:

$$[R_{yy}][C]_{opt} = [R_{ys}] \tag{1.8}$$

$$[C]_{opt} = [R_{yy}]^{-1}[R_{ys}] \tag{1.9}$$

where
$$R_{ys}(\Delta) = E[y(k)s(k+\Delta)] \tag{1.10}$$

and
$$R_{yy}(\Delta) = E[y(k)]y(k+\Delta)] \tag{1.11}$$

where $C_{opt}$ denotes the column vector of $2K+1$ optimal tap weight coefficients, $R_{yy}$ denotes the $(2K+1) \times (2K+1)$ co-variance matrix of the received vector of $(y_k)$, and $R_{ys}$ is a column vector of $(2K+1)$ with the correlation between received vector of $(y_k)$ and transmitted symbols $(s_k)$. $R_{ys}$ and $R_{yy}$ are estimated from a block of $10^5$ noisy bits. The estimated optimal tap weights (Eq. 1.9) are convolved with the received symbols, and the output is the equalized symbols.

The nonlinear optimal MLSE equalizer can be implemented in MATLAB using the built-in mlseeq command, with the inputs being: received sequence $(y(k))$, exact channel response used in simulations, and the trace-back length. The trace-back length is defined as the sequence length after which the algorithm traces back through trellis and outputs the most likely transmitted sequence. We simulate the optimal MLSE performance as we have perfect CSI. Practical systems would see this performance decreased when CSI is imperfect.

### 1.3.4 BER performance of optimal equalizers

In this section we compare the performance of optimal receivers for high baud rate transmission in a 100 Gbaud optical SiP channel (introduced in Section 1.2.2), for QPSK.

In Fig. 1.5, we report Monte Carlo simulations of BER vs SNR performance of MMSE and MLSE equalizers, estimated over $10^5$ symbols. For a high baud rate transmission, the BER curves of a theoretical channel ($\infty$ BW) with no-ISI, and a bandlimited optical SiP channel without equalization are provided as benchmarks. The MLSE and MMSE performance are

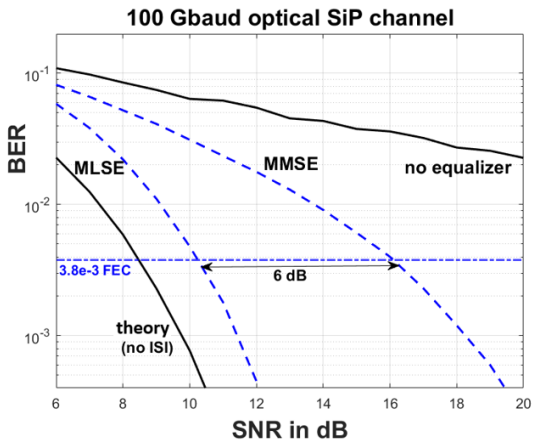



Figure 1.5 – BER performance of optimal receivers for 100 Gbaud optical SiP channel.

given in blue dashed lines. We can see that both equalizers improve the BER performance compared with case without an equalizer. Being a sub-optimal solution, the linear MMSE equalizer shows less BER improvement compared to that of nonlinear optimal MLSE equalizer with perfect CSI; this leads to a 6 dB performance gap.

We expect that as ISI severity increases, the MLSE-MSE performance gap would increase. In Chapter 3, we introduce several synthetic bandlimited channels, allowing us to sweep through the ISI severity. We will use this MLSE-MMSE gap as a quantifier of ISI impact. We will examine the relative performances of NNs and conventional equalizers by sweeping through the ISI severity and modulation order. In the next chapter, we provide some background knowledge about the machine learning techniques with the focus on NNs, to understand our simulations and NNs performance in Chapter 3.

# Chapter 2

# Machine Learning Techniques

Machine learning (ML) techniques [17] are not new, and many of the algorithms driving today's applications have been around for decades. In 1959, Arthur Samuel, a pioneer in the field of computer gaming and AI, first introduced the term "Machine learning" [30]. Interest in pattern recognition continued into 1970s, as described by Duda and Hart in 1973 [31]. Later, some major advances took place along the time; a famous paper on back-propagation (BP) by David Rumelhart, et al. made the BP work faster than earlier approaches to learning [32]. It made NNs to solve problems which had previously been insoluble. Today, the BP algorithm is the workhorse of learning in neural networks (NN) [33; 18]. In early 2000s, the availability of large data sets, increased computational power and development of cutting-edge models, made ML technology more popular and successful.

In principle, ML is the science of getting computers to realize a task without being explicitly programmed. Unlike classical DSP algorithms, ML methods do not need either exact or complete knowledge to accomplish the task. Instead, they use statistical techniques to learn the missing information and find the approximate fit of the model to the data. In particular, supervised learning methods use the pairs of inputs and desired outputs to learn the input-output mapping in a data driven fashion. Among supervised learning algorithms, NNs have received most attention; they have shown remarkable success in the fields of signal processing [22; 23]. In our work, we do not claim any fundamental contributions to ML, but rather use it as a toolbox. More precisely we introduce an overview of important concepts in ML, that aids the reader to understand our results in Chapter 3.

The rest of this chapter is organized as follows. Section 2.1 describes the basic building blocks of NNs. We focus on the structure of a single neuron, the role of activation functions and the error criteria used to train our NNs. In Section 2.2, we introduce the gradient descent and BP optimization algorithms for NN training; learning curves for monitoring the NN performance are also described. Later in Section 2.3, we provide the literature review of NN architectures for ISI channel equalization. In Section 2.4, we train our first baseline linear feedforward

NN (LFFNN) architecture for ISI mitigation, that could achieve only MMSE performance. In the last section, we emphasize that deep learning architectures are required for sequential processing tasks by providing a few examples from natural language processing.

## 2.1 Neural Networks (NN)

The idea of an artificial neural network (ANN) or neural network (NN) [33; 18] is inspired from the biological neural structures in the human brain. The human brain is the most powerful information processing device. It performs complex operations like vision, speech recognition and learning through neural connections in the brain. NNs attempt to simplify and mimic this brain behavior. They try to learn complex relations using artificial neural connections, and are built of many nodes called neurons. Each artificial neuron has an ability to input, process and output information. These neurons are grouped together to form a layered NN architecture, where outputs of the previous layer serve as the input for the next layer and can form deep structures.

### 2.1.1 Structure

#### Neuron

A neuron is the basic building block of NNs. In Fig 2.1, we show the typical structure of a single neuron. It consists of $D$ inputs that are each multiplied by a different synaptic weight. These weights decide the significance (firing strength) of each connection. An activation function ($f$) is applied to the linear combination of these weighted inputs and results in the output. The input-output relation for a single neuron can be formulated as:

$$y = f\left(\sum_{i=1}^{D} w_i x_i + w_0\right); \tag{2.1}$$

where $x_1$, $x_2 \ldots$. $x_D$ denote the input vector, $y$ is the output, $f$ is the activation function, $w_1$, $w_2$,...,$w_D$ are the weights, and $w_0$ is the bias (offset of the node activation). A layer is

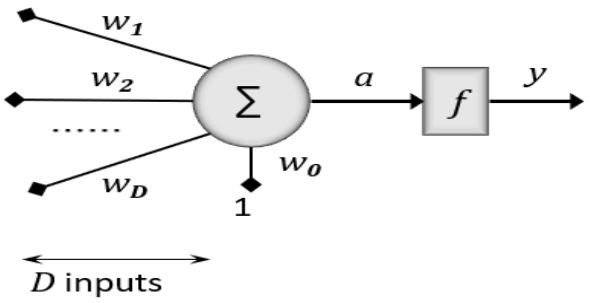



Figure 2.1 – Single neuron.

formed by multiple neurons (nodes) in parallel. Thus, each neuron in a layer consists of the same input vector but a different weight vector. Hence each neuron has different significance (given the same input vector) and fires independently to predict the output.

### Activation function

The activation function can also be described as a "mathematical gate" between the weighted inputs feeding the current neurons and its output going to the next layer. Usually a non-linear activation function is used in NN training, that helps to better model the complex input-output relations [34]. Activation function can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold. ML libraries offer a variety of non-linear activation functions like ReLU, Sigmoid, Softmax, tanh, etc. We use the tanh activation function for our NNs, as it yielded a higher performance in training compared to Sigmoid an ReLU activation functions. The tanh function is given by

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

The major advantage of a tanh activation function is that it has the output range between (-1,1). It allows for efficient training when output targets are both negative and positive (such as in QAM). This makes the NN training easier by mapping the inputs to the outputs that can have either strong negative or positive or neutral values.

### Neural Network architecture

A single neuron can take any value in weight-space. With different activation rules, it can represent a wide range of functions. The NNs, which can come in a myriad of forms, can thus approximate any arbitrary, complex functions [35]. They are also termed Universal function approximators. A typical NN is shown in Fig. 2.2, in this case with an input layer of 5 inputs, two hidden layers (note that NN can have one or more layers), and an output layer with 2 output nodes. The input layer takes the external data into the network and has one neuron for each input component also termed as "input feature" which are forwarded to (one or more) hidden layers present in the network. Finally, the processed data in the last hidden layer is sent to the output layer that has one neuron for each possible desired output.

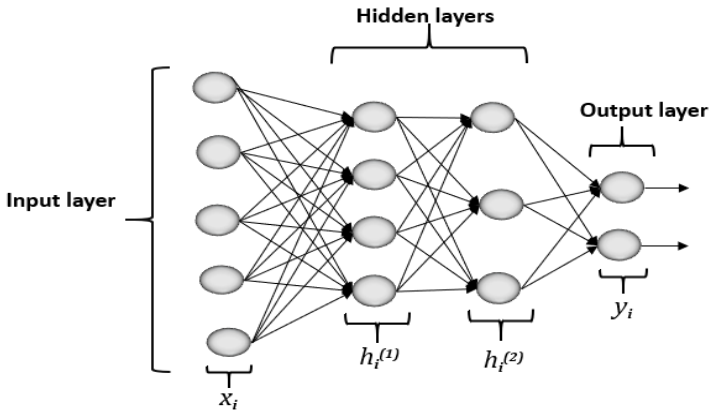



Figure 2.2 – Typical 2-layer NN architecture.

This NN can also be described by the following set of equations for our example in Fig. 2.2.

$$h_i^1 = f\left(\sum_{j=1}^{5} w_{i,j}x_i + w_{i,0}\right); \tag{2.3}$$

$$h_i^2 = f\left(\sum_{j=1}^{4} w_{i,j}h_i^1 + w_{i,0}\right); \tag{2.4}$$

$$y_i = \sum_{j=1}^{3} w_{i,j}h_i^2 + w_{i,0}; \tag{2.5}$$

where $\{x_i\}$ represent the input features and $\{y_i\}$ represent the output nodes. The hidden nodes of layer L are $\{h_i^L\}$ determined by their $\{w_{ij}^L\}$ (weight matrices) that connect $L-1$ layer to $L$ layer. Increasing the number of hidden layer nodes increases the number of free parameters (weights), which allows the NN to better approximate arbitrary complex functions. Increasing the number of hidden layers, on the other hand gives the NN the ability to represent complex hierarchical relations. The number of nodes and hidden layers determine the computational complexity of a NN. A very complex architecture can be prone to over-fitting (described in section 2.2.5). During NN training, a simple and yet better performing NN architecture is often the objective.

### 2.1.2 Regression vs. classification

The primary task of the NN is to approximate the mapping function from input variables $\{x_i\}$ to output variables $\{y_i\}$ . If the NN must predict a continuous output variable, i.e., real-value (such as an integer or floating-point value), the task is called a "regression". In contrast, the task of predicting a discrete output variable, such as labels or categories for a given input observation, is termed a “classification”.

In the context of this project, a NN used to classify and detect the received symbols after

transmission through a bandlimited channel is called a classifier. A NN used to mitigate ISI and predict the equalized IQ coordinates, i.e., real values, is termed a regressor. In the case of a regression, a simple linear output layer with two nodes delivers the equalized I & Q values (as in Fig. 2.5). In the case of classification, the size of output layer is $M$, i.e., the symbol constellation size. In this case, a soft-max activation function is used at the output layer to predict the symbol (class) probabilities and then to decide on the transmitted symbol.

### 2.1.3 Error criteria

In training the NN, the samples of the training set are used to adjust (learn) the NN weights, such that the predicted output becomes similar to the target (desired output). This similarity, or the closeness, is quantified by an error criteria, usually based on the required outcome of the task. The desired error criteria for our task (as BER) is highly nonlinear and may not be differentiable. In such cases, a "proxy error function" is usually minimized, leading to the desired error to also tend to a minimum.

In our regression NNs, we minimize the mean square error (MSE) value between the output and the target. The cross-entropy (CE) is used for our classifier NNs. The following equations give the NN criteria for MSE and CE error, that are minimized during training:

$$e_{MSE}(w) = \frac{1}{2}\left[\sum_{i=1}^{2}(|t_i - y_i|)^2\right]_{(w)} \tag{2.6}$$

$$e_{CE}(w) = -\left[\sum_{i=1}^{M}(t_i)(\log(y_i))\right]_{(w)} \tag{2.7}$$

where $e(\cdot)$ is the error value for $w$ set of NN weights, $\{t_i\}$ and $\{y_i\}$ represent the target and output nodes, and $M$ is number of output nodes in NN classifier.

## 2.2 Optimization Algorithms for NN training

In this section, we describe the algorithms used to optimize NN weights. With a gradient descent optimization algorithm, NNs learn their weights by minimizing the error function, given the training data.

### 2.2.1 Gradient descent algorithm

The gradient descent algorithm is an iterative process that leads to the minimum of the error function (see Fig. 2.3). It uses the gradient (with respect to $w$) of the error function to make an informed step change in $w$. With the error gradient information, the current set of weights

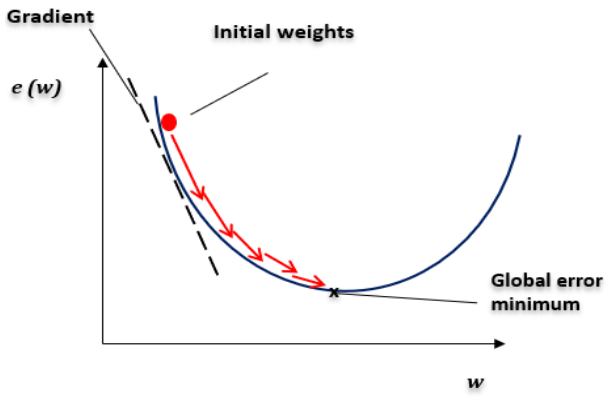



Figure 2.3 – Gradient descent method.

are updated per the equation below, such that the error is minimized.

$$w^{(l+1)} = w^{(l)} - \alpha \nabla_w e(w^{(l)}) \tag{2.8}$$

where $\alpha > 0$ is the learning rate, $l$ is the number of the training step iteration, $\nabla e(.)$ is the gradient of the error function, and $w^{(l)}$ is the weight matrix at $lth$ iteration.

During training, as the NN approaches the minimum of the error function, only small error improvements are made, due to the decrease in gradient value (Fig. 2.3). This results in flattening out of the error. This can also be interpreted as the NN convergence, where NN has achieved its minimum error and cannot further improve. Upon convergence, the iterative training process can be exited using a stopping criteria. We can implement the stopping criteria either by limiting the number of training iterations (as in this work) or by simply exiting the loop when the error drops below a certain limit (often called the precision).

When using gradient descent algorithm, the learning rate parameter must be tuned, which involves certain challenges. A learning rate that is too small leads to painfully slow convergence, while a learning rate that is too large can hinder convergence and cause the error function to fluctuate around the minimum, or even diverge. The error function demonstrated in Fig 2.3 is a simplex convex function that is easier to optimize. However, in deep NNs involving complex functions or noisier data, the error surface is usually very rough or highly non-convex. This can trap the algorithm into sub-optimal local minima.

### 2.2.2 Stochastic vs. Batch gradient descent

An epoch is one complete presentation of the data set to be learned by NN. A process is called batch gradient descent, if the entire training set is used to compute the error. In contrast, a stochastic gradient descent (SGD) [18; 36], updates with smaller batches that are randomly selected from the subset of data. It can be regarded as the stochastic approximation of gradient descent optimization, since it obtains an estimate of gradient (calculated from

randomly selected data subset) rather than computing the actual gradient (calculated from entire data set). Thus, SGD adds uncertainty to the error calculation and thus to the training.

SGD also eases the memory demands of the gradient descent optimization, by computing gradients from smaller batches. It is therefore usually much faster and can also be used to learn in real time. A batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces. SGD's fluctuation in training in contrast, enables it to jump to new and potentially better local minima. Momentum is usually added to SGD optimization to improve the convergence and reach global minimum [37]. Considering the less computational burden and faster convergence, we use SGD with momentum as our optimization algorithm for ISI equalization in QPSK.

### 2.2.3 SGD with momentum vs. Adam optimization algorithm

The error surfaces for deep learning models are usually very rough and non-convex [38]. Using an SGD optimization algorithm with added momentum allows us to minimize the error fluctuations and increase the rate of convergence. Difficulty arises when SGD is navigating error areas where the surface curves are steeper in one parameter dimension than in another (which are common around local minima). Additionally, it applies the same learning rate to all parameters. If we have sparse data, i.e., features occurring at very different frequencies, we might not want to update all parameters to the same extent, but rather perform a larger update for rarely occurring feature.

To overcome all such difficulties, advanced versions of SGD optimization are used that make the step updates using either adaptive learning rates or momentum. Adaptive Moment Estimation (Adam) [39] is one such popular method that computes the individual adaptive learning rates for different parameters from the estimates of the first and second moments of the gradients. These per-parameter learning rates can improve the performance on problems with sparse gradients (e.g., NLP and RNNs). In our work, we use Adam for higher QAM equalization (where error surfaces are more rough), as it showed better performance than SGD with momentum.

### 2.2.4 Back-propagation algorithm

In the previous sections, we discussed the learning of weights and biases of NN, using a gradient descent algorithm. In this section, we briefly discuss the computation of the gradient loss function. The previous gradient computation in the delta rule (in Eq. 2.8), is applicable for a single layer NN. But in a multi-layer NN, the output of hidden layer nodes are not directly connected to the error function, but affect the error function through mediating weights and potentially other layers of nodes. In that case, the variations (gradient) in the error function with respect to variations in the weights embedded deep within the NN are efficiently computed using the back-propagation (BP) algorithm [33; 18; 32].

The BP algorithm generalizes the delta rule and efficiently computes the gradients. It works by computing the gradient of the error function with respect to each weight by the chain rule, computing the gradient one layer at a time. It iterates backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule. This efficiency makes it feasible to use gradient methods for training multi-layer networks.

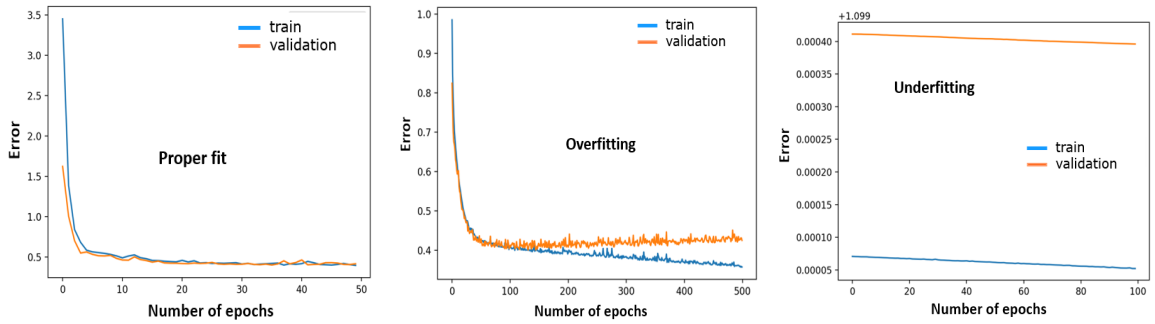
Advanced, or deep, NNs can have thousands of learnable parameters and computing these gradients at each layer in the backward pass using the chain rule is a bit complicated. Pytorch, like most other deep learning libraries, computes the partial derivatives of these functions using a numerical method called as automatic differentiation [40; 41]. This method is similar to the BP algorithm for differentiating NNs, but more general. It exploits the fact that every mathematical model, such as NNs, can be described by a sequence of elementary operations and functions. Automatic differentiation applies the chain rule repeatedly to sequences of operations and functions. Thus the gradients of the error function can be computed automatically, accurately and quickly compared to other classical methods. The gradients are then used for optimization. In our research, we use Pytorch's automatic differentiation engine, called "Autograd" to automatically calculate the gradients during training.

### 2.2.5 Network learning curves

During the NN training it is very crucial to monitor the performance of both training and validation data sets, to see if the model is truly generalizing for the new data. Usually the initial data set is split into three: training, validation and test sets. We use the training set to train the model, and the validation set to assess performance in the static NN. Once the model is completely trained, we use the test data (new unknown data) for assessing performances.

The error of the trained algorithm on the validation set reflects its performance on unseen data during the training process. During training, we calculate the error performance on both training and validation sets and monitor this performance improvement using learning curves. A learning curve is a plot that shows time or epochs on the x-axis and error performance or model accuracy on the y-axis. A model is said to be well generalized on new unseen data, if the difference between the training and validation error is very low during training (Fig. 2.4 (a)). For a good model fit, this error value (on y-axis) decreases with the advancement in training or increase in epochs (x-axis). We continue to train the model with respect to epochs until there is no further improvement in the error performance, i.e., until the NN converges.

In some cases, validation error continuously increases with new weight updates (and never decreases again) while the training error still keeps decreasing, see Fig. 2.4 (b). This is called overfitting of the model. In that case, the algorithm is not learning but fitting to unnecessary patterns in training data. This occurs when models, during training, become too complex. They adapt well to training data, but perform poorly on unseen data. In those cases, to avoid




(a) (b) (c)

Figure 2.4 – Neural network learning curves for (a) proper fit, (b) overfitting, and (c) underfitting.

overfitting, regularization techniques can be employed with error criteria. This reduces the overall model complexity, along with minimizing the error. In contrast to overfitting, there is another phenomenon called underfitting. In this case, both training and validation error are saturated, i.e., do not improve after a certain point or increasing epochs, see Fig. 2.4 (c). It occurs when the model complexity is too low for predicting the training or validation output properly. In this case, the model complexity must be increased to improve the performance. Using learning curves, we can sweep the NN hyper-parameters and choose a better performing NN configuration to avoid overfitting and underfitting.

## 2.3 Literature Review of NNs for ISI Mitigation

In this section we provide a brief overview of NN architectures previously used for ISI channel equalization and outline important results. We discuss briefly open problems from the literature that are targeted in this work.

In [42], authors implemented an artificial NN (ANN) to mitigate ISI for M-QAM transmission, and achieved BER improvement compared with conventional OFDM systems without any equalizer. This established that NNs can reduce ISI. Multi-layer perceptron (MLP) and radial basis function (RBF) NNs are proposed as two methods for nonlinear ISI mitigation in [43]. In [44] a reduced decision feedback Chebyshev functional link ANN (RDF-CFLANN) is implemented. These architectures could achieve only a small BER gain over a linear equalizer. In [45], a recurrent NN (RNN) is implemented with an M-level sigmoid function in a single output node for M-PAM detection in the presence of ISI. This structure reduces the output layer complexity, but does not show any significant performance gain. For mitigating nonlinear ISI, in [46] authors employed a ANN with nonlinear equalizer (NLE) neurons. In [47] nonlinearity is introduced in the NN as a trigonometric polynomial expansion. All the previously mentioned NN techniques could only outperform the linear equalizer by a minor improvement ($<$ 2 dB) for low ISI channels.

In [48], a near optimal MLSE solution is achieved using a iterative MLSE approach, however its complexity grows exponentially with channel memory length. In [49], authors proposed ViterbiNet where they estimate weights of the MLSE Viterbi trellis using a deep neural network (DNN). This model performs well when the CSI is unavailable. But like MLSE, ViterbiNet complexity grows exponentially with both modulation order and memory length. In [50], a DNN is used to combat dispersion in optical fibers, however, they do not address bandlimited channels. Recently, in [51] a recurrent neural network (RNN) architecture is used to mitigate ISI induced by a Poisson channel with the performance approaching MLSE. However, a Poisson channel represents a very low data rate channel and is a poor model for high data-rate optical communications.

In a nutshell, the previous works could demonstrate improved BER performance over a linear solution, while the solutions targeting optimal MLSE have difficulties with computational scaling like MLSE. Furthemore, those NN solutions did not target mitigation of severe ISI induced by the band-limitation at higher QAM. In this work, we develop NN architectures to mitigate severe ISI and also compare our NN performance to that of conventional linear and nonlinear optimal receivers. In the next section, we train our simple baseline NN and demonstrate its BER performance for a 100 Gbaud optical SiP channel. In Chapter 3 we introduce our NNs achieving optimal MLSE.

## 2.4 Training a NN for ISI Mitigation

To train a NN for ISI mitigation, we first generate our random M-QAM modulated training and validation data sets, and transmit it through our simulated channel that adds ISI and AWGN. We input received symbols and targets symbols to the NN. After training the NN it outputs the equalized symbols. The equalized QAM symbols are demodulated using standard decision boundaries and the output BER is calculated. Further details on NN training and simulations are provided in the Appendix.

### 2.4.1 Linear feedforward NN (LFFNN) for ISI mitigation

In this section, we first train our simple baseline NN equalizer - a linear feedforward NN (LFFNN) using the MSE criteria. We model this NN as a regressor for demonstration, however, LFFNN has similar performance for classification. SGD is used as the optimisation algorithm, while the weights are updated using the back-propagation algorithm. Note that for the more complex architectures in Chapter 3, we use the more classic CE criteria to include symbol decisions with the equalization function. We also switch to Adam optimization for better convergence.

This simple LFFNN provides a baseline and has input and output layers and one hidden layer with 70 linear neurons. The input layer has 58 features: in-phase (I) and quadrature (Q)

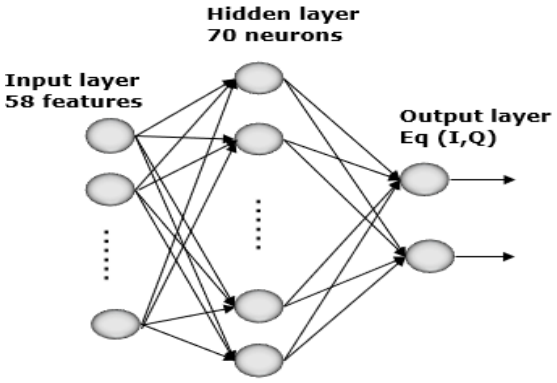



Figure 2.5 – LFFNN architecture.

components of the targeted symbol, as well as 14 past and 14 future symbols. The LFFNN weights are adapted to minimize the MSE of the training data. The input data is then convolved with the final LMLP weights to give the equalized channel outputs.

In Fig. 2.6, we report BER vs. SNR performance of LFFNN equalizer, estimated over $10^5$ symbols for 100 Gbaud optical channel (introduced in Chapter 1) for QPSK transmission. In Fig. 2.6 we reproduce the BER curves for MMSE, MLSE equalizers for QPSK transmission in Fig. 1.5, adding LFFNN results. From Fig. 2.6, we observe the LFFNN (diamond markers) attains the performance of the MMSE equalizer. This shallow (single layer) LFFNN architecture without nonlinear operations and without a feedback path, unsurprisingly achieves only MMSE performance, but not MLSE. To target optimal MLSE performance, we explore deep learning (DL) architectures in Chapter 3, that hold the potential to effectively mitigate ISI sequential distortion. In the following section, we briefly introduce the DL architectures and motivate their use by providing their recent successes in the field of natural language processing; the word sequence correlations are not unlike the pattern dependencies in ISI distorted data.

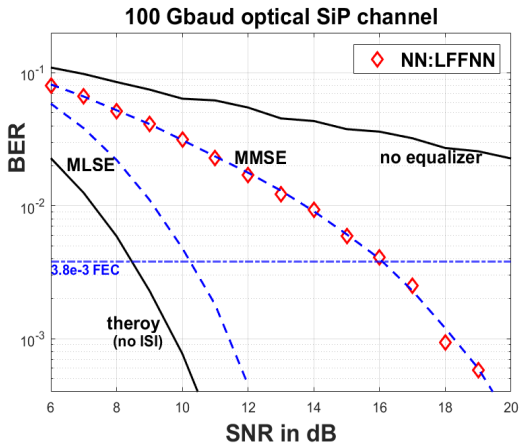



Figure 2.6 – BER performance of LFFNN for 100 Gbaud SiP channel.

## 2.5 Deep Learning Architectures

A deep neural network (DNN) commonly has between 2-8 additional hidden layers of neurons, in contrast to a shallow NN with only one hidden layer to process the inputs. However, deep learning (DL) models do not necessarily include only deep architectures (with many layers), but also convolutional NNs (CNN) [52; 53] and recurrent NNs (RNN) [18; 19]. CNNs are most suitable for visual imagery tasks such as face recognition where the surroundings pixels are correlated. For sequential processing tasks (like ISI mitigation), RNNs use their inherent feedback path to analyze the hidden sequential correlations in the data.

The key aspect in DNNs is obtained by composing simple but non-linear modules, that each transform the initial representation into a different and more useful representation, i.e., features. The features learned are not designed by human engineers, but are gleaned from data using a general-purpose learning procedure. DL models are intrinsically much more powerful than shallow circuits. These methods overcome the drawbacks of shallow ML methods such as time-consumption and hand-crafting features. Furthermore, the implementation of these learning algorithms on a graphics processing unit (GPU) architecture increased the training speed of DNNs and hence the amount of data it can be trained on.

In particular for the sequential processing tasks in nature language processing, different combinations of deep, convolutional and recurrent NNs are very effective. In [54], a simple DNN with extracted framed input features is used for speech activity recognition in YouTube. In [55] for speech recognition, authors implemented a CNN followed by RNN for better capturing the temporal patterns. Their results show that CNN enhanced the feature extraction capability of RNN. For action recognition in video sequence in [56], authors proposed the use of deep bidirectional long-short term memory (LSTM) with CNN extracted features. The long-short term memory (LSTM) [24] is an advanced version of RNN, that has a complex gating operation to efficiently capture long-term sequential dependencies.

Motivated by these architectures for sequential processing, in the next chapter we develop our NNs for severe ISI mitigation: a feedforward NN with a framed input and also two LSTM architectures with and without framed input. We examine the performance of these NNs by sweeping the number of input features, hidden nodes, depth and the error criteria.

# Chapter 3

# Recurrent Neural Networks Achieving Optimal MLSE Performance for Optical Channel Equalization

## 3.1 Résumé

Nous explorons les réseaux neuronaux à rétroaction et à action directe pour atténuer les interférences intersymboles graves (ISI) causées par les canaux à bande limitée, tels que les systèmes de communication optique à grande vitesse poussant la réponse en fréquence des composants de l'émetteur. Nous proposons une nouvelle architecture de mémoire bidirectionnelle profonde à long terme (BiLSTM profond) qui met fortement l'accent sur les dépendances dans les séquences de données. Pour la première fois, nous démontrons par simulation que pour la transmission MDPQ, le BiLSTM profond atteint le taux d'erreur sur les bits optimal d'un estimateur de séquence à maximum de vraisemblance (MLSE). Nous évaluons la performance pour une variété de canaux présentant une ISI, y compris un canal optique à 100 Gbaud utilisant un modulateur photonique au silicium (SiP) de 35 GHz. Nous montrons comment la performance du réseau neuronal diminue avec l'augmentation de l'ordre de modulation et de la sévérité de l'ISI. Bien qu'il n'atteigne plus les performances du MLSE, le BiLSTM profond surpasse largement l'égalisation linéaire dans ces cas. Plus important encore, le réseau neuronal ne nécessite aucune information sur l'état des canaux, alors que ses performances sont comparées à celles d'égaliseurs ayant une connaissance parfaite des canaux.

## 3.2 Abstract

We explore recurrent and feedforward neural networks to mitigate severe inter-symbol interference (ISI) caused by bandlimited channels, such as high speed optical communications systems pushing the frequency response of transmitter components. We propose a novel deep

bidirectional long short-term memory (BiLSTM) architecture that strongly emphasizes dependencies in data sequences. For the first time, we demonstrate via simulation that for QPSK transmission the deep BiLSTM achieves the optimal bit error rate performance of a maximum likelihood sequence estimator (MLSE). We assess performance for a variety of channels exhibiting ISI, including an optical channel at 100 Gbaud operation using a 35 GHz silicon photonic (SiP) modulator. We show how the neural network performance reduces with increasing modulation order and ISI severity. While no longer achieving MLSE performance, the deep BiLSTM greatly outperforms linear equalization in these cases. More importantly, the neural network requires no channel state information, while its performance is compared to equalizers with perfect channel knowledge.

## 3.3 Introduction

To meet rapidly growing traffic demand, optical communications systems are turning to advanced modulation formats combined with coherent detection. The electrical bandwidth limitations of the transceiver components pose the major challenges in achieving higher data rates. Inter-symbol interference (ISI) due to this band-limitation, rather than signal-to-noise ratio, can be the principal impairment in higher order QAM modulation in high-speed optical communication.

ISI can be mitigated via post compensation of the received signal. The maximum-likelihood sequence estimator (MLSE) is an excellent solution to combat ISI, providing the optimal performance by finding the most probable transmitted sequence. However, an MLSE is highly complex and becomes infeasible with increasing modulation order and ISI memory length. Moreover, to achieve this optimal performance, the MLSE equalizer requires accurate channel state information (CSI). Research focus has thus shifted towards sub-optimal solutions. The minimum mean squared error (MMSE) equalizer provides an optimal linear solution. However, the MMSE performance quickly deteriorates with severe ISI.

Recently, machine learning [17] and deep learning [18; 19] techniques have been applied in many areas of communication [20; 21; 22; 23]. Neural networks (NN) hold the potential to learn the channel indirectly from the data during the training, without the need for explicit CSI. In [49], the authors estimate the weights of the Viterbi trellis for the MLSE using a deep neural network (DNN). This model performs well for the unknown channel conditions, but like MLSE, its computational complexity grows exponentially with modulation order and channel memory length. In [50], a DNN combats dispersion in optical fibers, however, they do not address the bandlimited channels. In [51], a recurrent neural network (RNN) architecture is used to mitigate ISI induced by a Poisson channel; the performance approaches that of MLSE. However, a Poisson channel is a poor model for high data-rate optical communications, and useful only for very low data rate channels.

We examine both recurrent and feedforward NNs to target high data rates in severely bandlimited channels. We propose the use of long short-term memory-RNN (LSTM-RNN) [24] for ISI, and extend examinations begun in [4; 3]. The recent success of LSTM-RNNs in speech recognition [57; 58; 59] and machine translation [60; 61] motivated our examination. The correlations in language across phrases, sentences and paragraphs is not unlike the pattern dependencies in ISI data. We show the deep bidirectional LSTM (BiLSTM) architecture is very promising for processing sequential dependencies. To the best of our knowledge, ours was the first demonstration of LSTM-RNNs achieving the MLSE performance in bandlimited channels with severe ISI [3] for QPSK. In this chapter we examine how modulation orders higher than QPSK leads to deep BiLSTM under perform MLSE, and how this behavior varies with the severity of the ISI.

We study two families of ISI channels, one multipath family and another super Gaussian family, to sweep through ISI severity. While these are synthetic channels, we also examine an experimentally measured optical channel from a silicon photonic (SiP) modulator [5] operated at 100 Gbaud.

The rest of the chapter is organized as follows. In Section II, we introduce conventional optimal equalizers, bandlimited channels (specifically those examined in our simulations) with their performance metrics, and traditional neural networks. In Section III, we introduce the deep bidirectional long short-term memory (BiLSTM) neural network and demonstrate that for QPSK modulation it outperforms other NN solutions, reaching MLSE performance. In Section IV we move to higher order QAM; we quantify performance penalties with increasing levels of ISI. In Section V, we provide some discussion of relevant NN characteristics, such as convergence and complexity. Concluding remarks are provided in Section VI.

## 3.4 System Description and Preliminaries

The functional block diagram shown in Fig. 3.1 illustrates the signal flow through a typical communication system. The modulated QAM signal is generated at the transmitter and passes through a channel with bandlimited components. At the receiver, additive white Gaussian noise (AWGN) corrupts the received symbols. When high data rate signals are transmitted through these bandlimited components, the signal is distorted due to severe attenuation at high frequencies. This results in severe intersymbol interference (ISI), even when using Nyquist pulse shaping to mitigate these effects.

To achieve reliable transmission, we can use post-compensation equalizers. In the next subsection we describe the optimal linear and nonlinear conventional equalizers. Following that we describe several collections of bandlimited channels that will be examined, and the motivation for their use. In particular, we quantify the conventional equalizer performance for these channels. We end this section with an introduction to feed-forward and recurrent neural networks

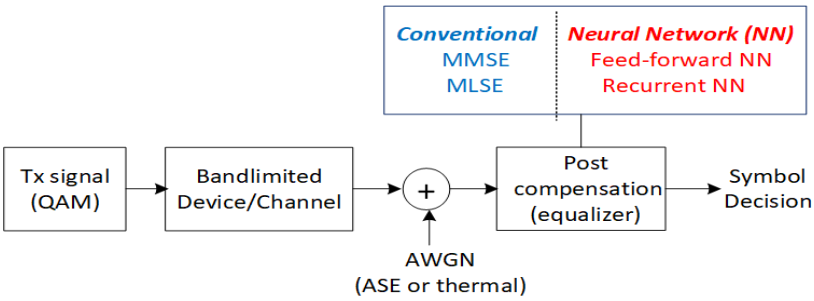



Figure 3.1 – Block diagram of simulated communication system.

(NN) as equalizers. Later sections discuss NN performance for the channels introduced.

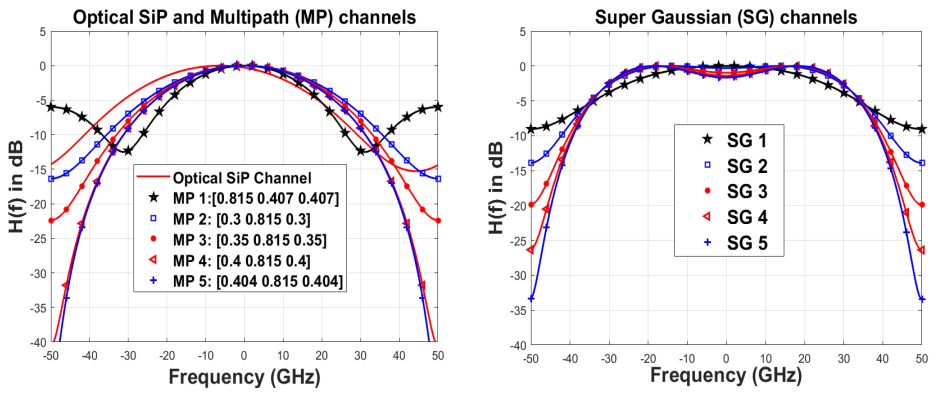
### 3.4.1 Conventional receivers

The minimum means square error (MMSE) receiver is the optimal linear approach to symbol-by-symbol detection to mitigate ISI. It is a model-based equalizer taking the form of a finite-impulse response filter. For a known channel, that is for perfect channel state information (CSI), the exact MMSE equalizer tap coefficients can be found. When CSI is not available, we can use data-driven adaptation of tap weights. The MMSE equalizer is relatively easy to implement and is an efficient solution for low ISI channels. Its performance is highly sub-optimal for compensating high ISI.

For an ISI channel the optimal nonlinear equalizer uses sequence detection rather than the symbol-by-symbol approach in a MMSE equalizer. It is known as the maximum likelihood sequence estimator (MLSE) equalizer. The MLSE is also a model-based equalizer, applicable to channels where a trellis-like architecture can describe symbol dependencies (as is the case for ISI channels) [16]. An exhaustive examination of all sequences is not necessary to find the optimal one, but the algorithm is highly complex and scales exponentially with both channel memory length and modulation order. Not only is prior CSI required, MLSE performance highly depends on the quality of the CSI. The MMSE and MLSE provide "bookends" to the performance/complexity trade-off in equalization.

### 3.4.2 ISI Channels for simulation

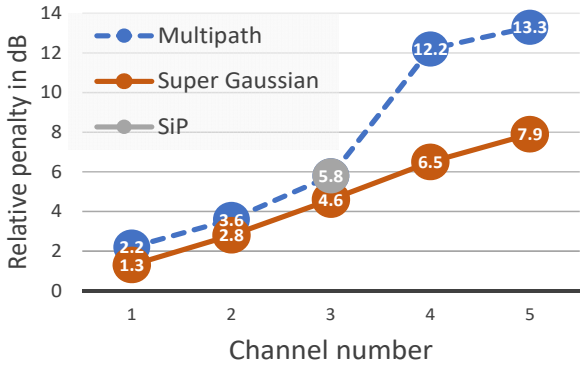
The impact of ISI on BER performance depends on two factors: the receiver used and the channel frequency response. We use BER performance of MMSE (linear) and MLSE (non-linear) receivers as a baseline of comparison with our proposed NN solutions. The relative performance of these two solutions (MMSE & MLSE) depends on the severity of the channel. In this subsection, we first introduce the collection of channels we simulate (including an experimental SiP modulator response), and secondly a means of quantifying the ISI impact of these channels vis-à-vis the performance baselines.




(a)




(b)




(c)

Figure 3.2 – (a) Optical SiP and five multipath channel frequency responses, (b) five super-Gaussian channel frequency responses, and (c) difference in MMSE and MLSE SNR penalty in dB for QPSK for various channels.

This work is motivated by the challenges in silicon-photonic (SiP) modulator operation at baud rates that greatly exceed their nominal channel bandwidth. Our simulations include the ISI created by a system with the experimental SiP modulator frequency response for 100 Gbaud operation [5]. The measured impulse response is complex and estimated to 512 taps. We truncate the taps to three, representing 90% of the total energy in the taps. This was done to facilitate comparison with other channels examined with similar tap lengths. The frequency response over a 100 GHz bandwidth is given in Fig. 3.2(a). The SiP modulator has a 3 dB bandwidth of 35 GHz.

Our neural net performance will fall somewhere between the MMSE & MLSE benchmarks. To generalize our conclusions we need to expand our examination to channels that can sweep through ISI severity. We investigate two series of synthetic channels responses. The first series is a collection of multipath channels with three taps. We have chosen the tap weights to cover various frequency responses, as seen in Fig. 3.2(a). We note that these multipath channels are variations on a classic example appearing in [14]. In Fig. 3.2(b) we plot five super

| Channel | Tap coefficients |
|---|---|
| MP1 | [0.815 0.404 0.404] |
| MP2 | [0.3 0.815 0.3] |
| MP3 | [0.35 0.815 0.35] |
| MP4 | [0.4 0.815 0.4] |
| MP5 | [0.404 0.815 0.404] |

(a)

| Channel | Tap coefficients |
|---|---|
| SG1 | [-0.0246 -0.2157 0.9517 -0.2157 -0.0246] |
| SG2 | [-0.0930 -0.2433 0.9297 -0.2433 -0.0930] |
| SG3 | [-0.1322 -0.2580 0.9121 -0.2580 -0.1322] |
| SG4 | [-0.1534 -0.2654 0.9011 -0.2654 -0.1534] |
| SG5 | [-0.1649 -0.2685 0.8952 -0.2685 -0.1649] |

(b)

Table 3.1 – Channel tap coefficients for (a) multipath (MP), (b) super-Gaussian (SG) channels

Gaussian channels (used extensively to model optical filters). Their taps take the form of a Gaussian exponential raised to a power (one to five in our parameterization). The channel tap coefficients for our examined channels are given in Table 3.1. While multipath channels have three tap weights, we use five weights for the super Gaussian channels.

All channel impulse responses are normalized to have unit energy. The bit error rate (BER) vs. signal-to-noise ratio (SNR) is easily found numerically for the MMSE & MLSE receivers in the case of perfect CSI. That is, performance when the receiver knows the exact channel impulse responses. Common forward error correction (FEC) techniques in optical communications systems are pegged to a FEC threshold of 3.8e-3 BER. Therefore each of our channels can be characterized by the SNR penalty of the equalizer vis-à-vis an ideal channel with no ISI. Figure 3.2(c) summarizes the gap between the SNR penalty for the MMSE and MLSE for QPSK modulation for the channels examined. We observe our selection spans a wide swath of penalties.

### 3.4.3 Supervised machine learning for equalization

In this chapter we investigate NNs as ISI channel equalizers, to overcome the limitations of conventional receivers, by providing a lower complexity and near optimal solution. We examine two NN architectures: a classic feed-forward NN (FFNN) and a recurrent NN (RNN). The

inspiration for the FFNN is to mimic the nonlinear nature of the MLSE, while the RNN seeks to mimic the sequence estimation.

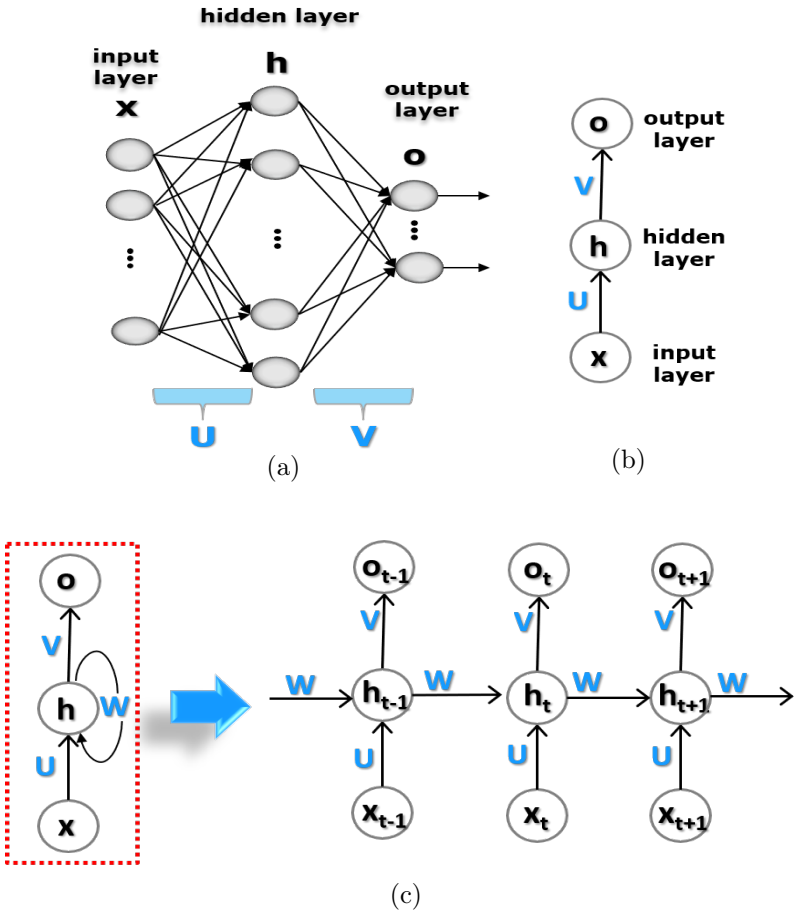



Figure 3.3 – (a) NN structure, (b) abstraction of NN operations, and (c) feedback introduced in RNN.

FFNNs are the simplest and most widely used NNs; information flows only in the forward direction. In Fig. 3.3(a) we show a typical FFNN with an input layer, a hidden layer (only one is shown, but multiple hidden layers can be used), and a final output layer. During NN training we examine various choices for the number of hidden layers, neurons and input frame size (features).

The input layer for our application is the received in-phase (I) and quadrature (Q) coordinates. The input could be a single pair of received IQ coordinates, or a frame of buffered IQ measurements in a sliding window. The input layer components are convolved with NN weights and sent to a hidden layer followed by a nonlinear activation function.

The NN weights can be trained to minimize mean-squared error (MSE) or cross entropy (CE) between the output and known target. When using MSE in a regression, we are refining the

IQ coordinates with the NN and then making a symbol decision; the output is equalized IQ coordinates. When using CE in a classification, the decision section of the receiver is part of the NN training; the output is the symbol decision. As is common with communications systems, we focus on classification.

When using the CE error criterion, the output layer has $M$ neurons, one per constellation point. We use the soft-max function at this layer to determine the symbol probabilities (class probabilities). We then apply the CE negative log likelihood error criterion for updating nodes.

In Fig. 3.3(b) we show an abstraction of the FFNN update. The input $x$ is acted on by the operator $U$ to produce the hidden states $h$. The state $h$ is acted on by the operator $V$ to produce the output $o$. The training from one input/output cycle is independent of other input/output cycles; i.e., there is no feedback and hidden layer updates are not influenced by previous FFNN hidden layer states. In this way, the FFNN output would resemble MMSE output; however, the FFNN introduces a nonlinear activation function while the MMSE is strictly linear.

To approach an MLSE solution, we turn to a recurrent NN (RNN). An RNN has feedback connections that could be used to capture the MLSE sequential processing behaviour. Correlations in QAM sequences are essentially treated like correlations in natural language processing, where RNNs have been shown to be extremely effective [57; 58; 59; 60; 61].

In Fig. 3.3(c) we show the abstraction for the update operations in a RNN with the $W$ operator representing the feedback. An additional memory cell in each hidden layer neuron provides access to the past state of the hidden layer. When the RNN updates are unfolded in time, see the right section of Fig. 3.3(c), we can visualize the influence on the output of previous hidden states (which in turn were influenced by previous inputs).

## 3.5 NN architectures for correlated sequences

Before moving to the next section to examine challenging higher order modulation, we simulate here the more robust QPSK format. We also confine our examination in this section to two channels: the SiP and a more severe ISI channel (MP5). We extend the basic architectural approaches (FFNN and RNN) introduced in the last section to NN architectures better suited to combat ISI.

### 3.5.1 Long short-term memory (LSTM)

Even a simple FFNN can have some information on sequential dependencies causing ISI by having the input layer include a sliding window (a frame) of IQ measurements. However, output computation is based only on the current input features and is independent of previous frames. An RNN can exploit both the frame and the previous hidden state. However, the RNN

quickly losses the impact of previous states, hence we consider the long short-term memory (LSTM) [24] version of the RNN. We will consider both the unidirectional and bidirectional LSTM versions.

The LSTM was introduced to better capture the very long-term dependencies. The simple RNN cells, which contained only the previous hidden state, are replaced by more elaborate cells, per the illustration in Fig. 3.4(a). The LSTM cells are multi-functional as seen in Fig. 3.4(b). The cell gates (in the form of sigmoid functions) perform complex operations on data (like forget, update and output) to capture the very long term dependencies during training.

A unidirectional LSTM will adjust to changes in the input sequence, even if a single IQ symbol is input. However, it cannot achieve the pruning action in a Viterbi algorithm [62]. An MLSE trellis [16] will update the past decisions (switch to a different path) at each successive symbol interval if that decision increases likelihood. This reevaluation of previously preserved paths is a sort of "back and forth" search for the best sequence in an MLSE trellis. If the RNN ran in two directions it would be able to harness previous paths via the RNN running in the reverse direction. We consider a bidirectional LSTM (BiLSTM) [63; 64] so that the forget and update functions can also be applied in reverse on the data.

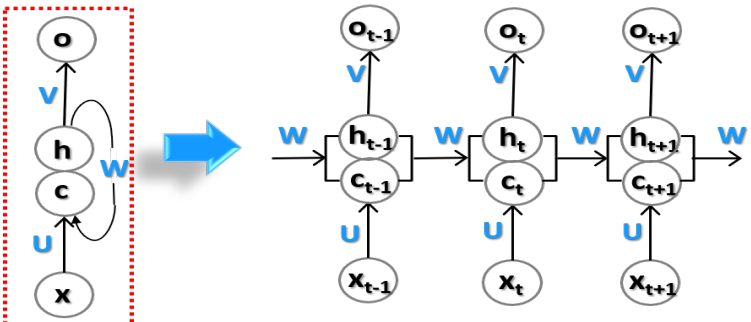
The abstraction of the BiLSTM architecture is shown in Fig. 3.4(c). It is formed from two independent and identical LSTMs. The input data is fed through each LSTM - one copy in the forward direction and one copy in the backward direction. The outputs at each symbol interval are then combined to create the final outputs. Thus, the equalized current symbol IQ coordinates at the output layer is calculated from both the past and future sequence information.

### 3.5.2 NN scenarios examined for QPSK

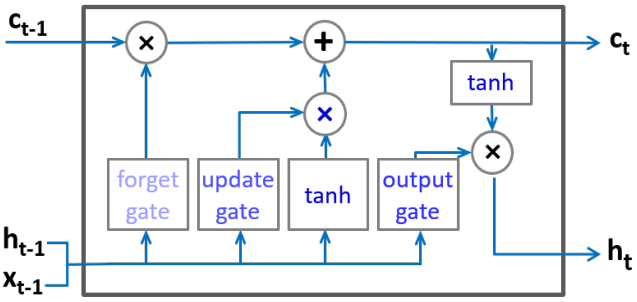
For QPSK we consider three NN approaches. In two approaches we provide the NN with the ability to observe the sequence via an input buffer with a frame of IQ measurements (the current IQ values are at the center of the frame). In the third approach we input only the current IQ measurement and rely on feedback mechanisms in NN to allow the weights to react to data sequences.

First we consider a FFNN with a frame input and nonlinear activation function. The second is the LSTM with a frame input. The forward-only LSTM has the feedback gating in addition to the frame input to react to data sequences. The third case is a BiLSTM with a single IQ input. The BiLSTM relies entirely on the LSTM cells (both in the forward and the backward NN) and their gating to gain context into symbol sequences.

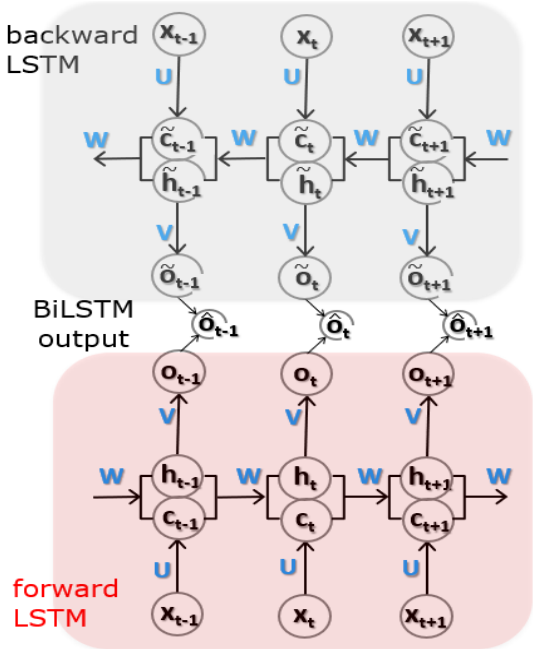
The FFNN and LSTM have input layers with 18 features (4 buffered past and future of the I




(a)

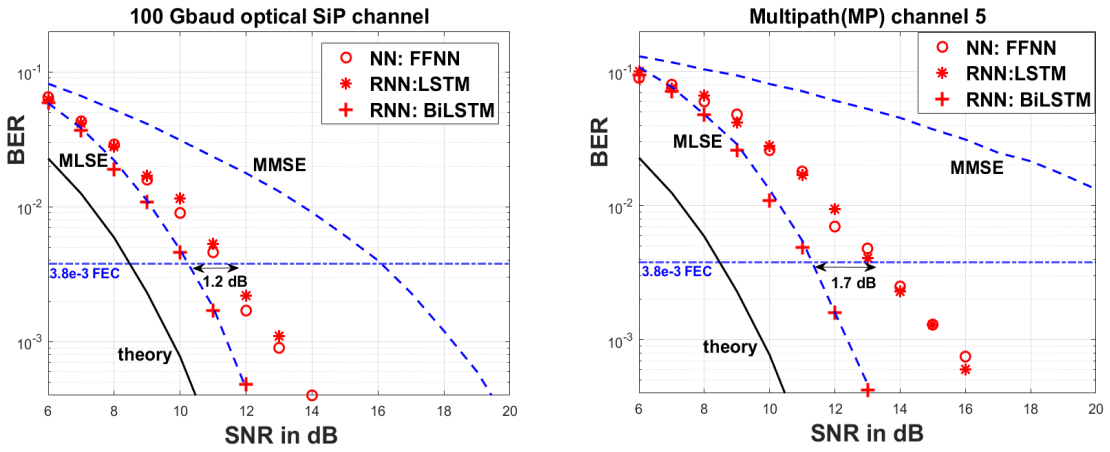



(b)




(c)

Figure 3.4 – (a) Abstraction of LSTM feedback operations, (b) LSTM cell label "c" in the abstraction, and (c) bidirectional version of LSTM.




(a) (b)

Figure 3.5 – QPSK BER performance of BiLSTM for (a) optical SiP channel and (b) multipath channel MP5.

and the Q measurements and the current I and Q measurement). Smaller frames led to worse BER, and larger frames did not improve BER. Given that the channels examined had only 3 or 5 taps in their impulse response, these values are not surprising. The MMSE solution used 31 taps in a linear filter, however, typically the vast majority of filter taps energy was in 17 taps (a main lobe was 5 taps wide with ∼50% of tap energy).

We use one hidden layer for LSTM, as a second hidden layer did not improve performance. For the FFNN we swept from one to four hidden layers. Performance improves for two layers, but remains flat for three and four layers. Therefore, we use two hidden layers for FFNN. For the BiLSTM architecture two hidden LSTM layers clearly improved performance, while a third did not. We refer to this NN as deep BiLSTM to highlight the depth as compared to the other LSTM solution examined.

The FFNN uses 50 nodes in each hidden layer, and the LSTM and deep BiLSTM both use 60 nodes (LSTM cells more accurately). We examined a range of 30 to 100 nodes for all NNs. We settled on 50 nodes for FFNN as this gave good performance at all SNR levels; more nodes gave no performance improvement. For LSTM and deep BiLSTM, 60 nodes was best.

We minimize the weights in all NN models via a stochastic gradient approach. To assist in convergence we use an adaptive gradient approach, specifically the adaptive gradient using the adaptive moment (Adam) optimization [39] with a learning rate of $\sim 10^{-2}$.

### 3.5.3 Performance results

We simulated BER vs. SNR performance for two channels: the SiP channel and the multipath channel MP5 which has considerably more severe ISI. The SiP and MP5 results are presented in Fig. 3.5(a) and (b), respectively. The BER for an ideal channel is included with the annotation

"theory". The conventional receiver performance, both MLSE and MMSE, is included in dashed blue lines. The NN performance is given in red markers. A horizontal line is traced at the 7% overhead FEC threshold at BER of 3.8e-3.

Consider first the SiP channel. The NN have performance that falls between the MMSE and MLSE receivers, that is, between the optimal linear and nonlinear solutions. The LSTM outperforms MMSE with a gain of 4.6 dB, with the FFNN providing similar improvement. At the 7% FEC threshold, LSTM approaches MLSE with a small 1.2 dB penalty. The deep BiLSTM equalizer actually achieves the same performance as the optimal MLSE receiver.

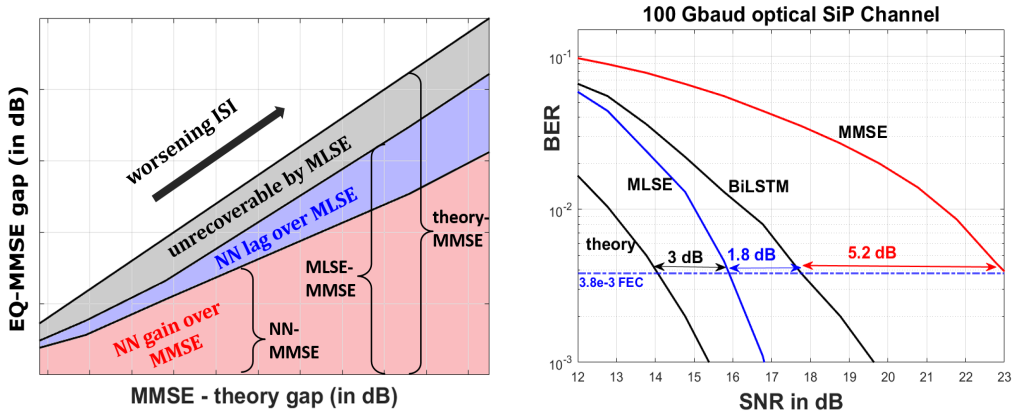
The same trends can be seen for multipath channel MP5. As the ISI is more severe, the disparity in the performance of the linear MMSE solution and the MLSE solution is much more pronounced. The penalty from MLSE to LSTM performance is only an additional 0.5 dB for this channel, despite a much greater gap between MMSE and MLSE performance (13.3 vs. 5.8 dB). Once again the deep BiLSTM achieves the same performance as the MLSE.

We evaluated the BER vs. SNR performance of deep BiLSTM for all our remaining synthetic bandlimited channels, i.e., all SG and all MP channels (given in Appendix A). At the 7% FEC threshold, the deep BiLSTM again achieved the same performance of MLSE, i.e., zero penalty. The deep BiLSTM is extremely effective in mitigating severe ISI for QPSK. This is especially remarkable as the MLSE had access to perfect CSI, while the deep BiLSTM garnered its information only from the training set.

### 3.5.4 Discussion

The classical FFNN and the LSTM offered similar performance. The LSTM had the same sliding window input as the FFNN, but even with the additional internal feedback it was unable to outperform the FFNN. We suspected that the framed input could actually be holding back the LSTM from even better performance. It was possible that providing the LSTM with too much context, stifled its ability to build context using the long short-term memory cells in the NN.

Confining the LSTM to only a single pair of IQ coordinates at the input did lead to mild improvement in performance. Moving to a bidirectional LSTM was required to reach the theoretical limit of MLSE detection. The deep BiLSTM architecture was effective in learning the sequential nature of data dependencies. This change, combined with the deep (two layer) structure, greatly enhanced the LSTM ability to address sequential correlations.




(a) (b)

Figure 3.6 – (a) BER performance gaps for prototypical evolution as channels worsen, (b) 8QAM BER performance for SiP channel.

## 3.6 Extension to Higher Order QAM

As NNs are data-driven models, their performance depends acutely on input data quality. Performance degradation with increased modulation order can be severe. In a channel with high ISI the received symbols (input data to NN) are highly perturbed by ISI distortion in addition to additive noise. We examine how ISI severity impacts the ability of the deep BiLSTM NN to achieve MLSE-levels of performance.

We use the same deep BiLSTM and BER simulator as previous sections, but replace QPSK with M-QAM modulation. We vary ISI severity by examining a variety of channels. The deep BiLSTM architecture (two hidden layers and 60 LSTM cells each) is unchanged. These parameters were varied, but we could not find a configuration with higher performance.

### 3.6.1 Assessing Performance Gains

Consider relative BER performance as a channel worsens, that is, as the ISI becomes more severe. The gap (in dB) between the MMSE and the ISI-free BER curves will increase. We can use the gap width at the 3.8e-3 FEC threshold as a benchmark. Each channel can be parameterized by this gap, which we use as the $x$-axis in the prototypical performance plot in Fig. 3.6(a). The $y$-axis reports the other performance gaps vis-à-vis MMSE, where 'eq' is one of the three equalizers: NN, MLSE, or theoretical limit (ISI-free). The lines report the absolute performance gap between MMSE and each solution (see curly braces).

The lines define three performance regions.

1. Unrecoverable performance with (nonlinear) MLSE is seen in
   - grey region in Fig. 3.6(a);
   - black annotation in Fig. 3.6(b) & Fig. 3.7.
2. NN performance lag over MLSE is seen in
   - blue region in Fig. 3.6(a);
   - blue annotation in Fig. 3.6(b) & Fig. 3.7.
3. NN performance improvement over (linear) MMSE is seen in
   - red region in Fig. 3.6(a);
   - red annotation in Fig. 3.6(b) & Fig. 3.7.

A large red region means the NN has made great gains over the linear MMSE. A small blue area means the NN is performing well compared to the optimal MLSE equalizer. In the ideal NN performance case the blue region disappears and the red region is maximized. A large grey area means the channel is truly challenging and even the optimal MLSE has limited performance. Note that the uppermost line is by construction $x = y$, the gap between MMSE-theory.

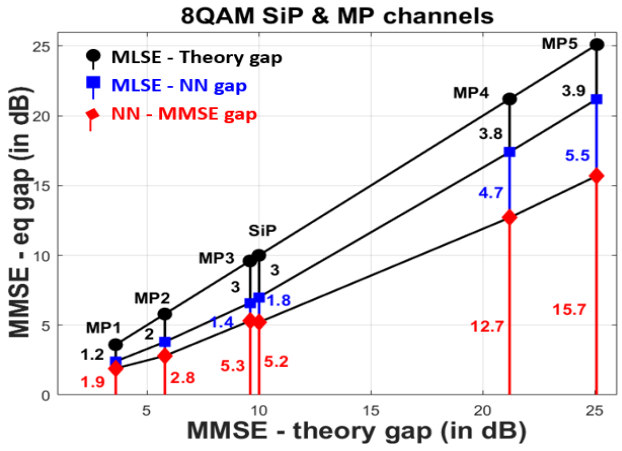
To move from the prototypical plot to a specific plot we run simulations of BER vs. SNR for each channel. Take the example in Fig. 3.6(b) for 8QAM over the experimentally measured SiP channel frequency response. We annotate the FEC line with the relative penalty between equalizers - deep BiLSTM (black), MLSE (blue) and MMSE (red). For the SiP channel, the gap at the FEC threshold between no-ISI and a linear MMSE equalizer is 10 dB.

### 3.6.2 Sweeping channels and modulation order

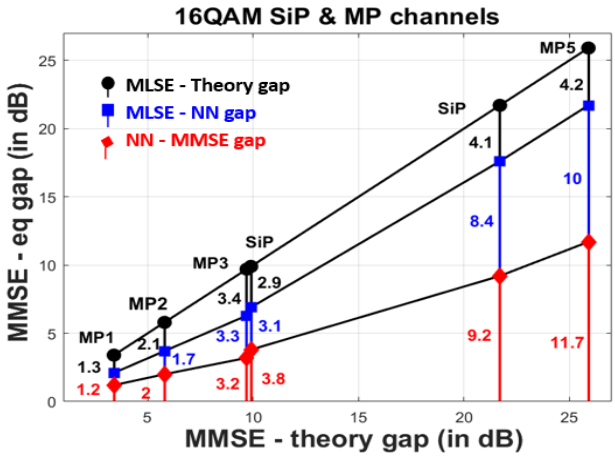
In Fig. 3.7(a) we present 8QAM results for SiP and five multipath channels described in section 3.4.2. The SiP stem plot is centered at $x = 10$ dB. Beside each section of the stem we note the corresponding relative penalty in dB found at the FEC threshold in Fig. 3.6(b). The other stem plots are found in a similar manner.

With this new graphical view of performance, we examine the effects of ISI intensity and QAM modulation order. In addition to the 8QAM case in Fig. 3.7(a), results for 16QAM and 32QAM are given in Fig. 3.7(b) and Fig. 3.7(c), respectively. We found the super-Gaussian channels follow similar trends to those of the multipath channels; see plots in section 3.7.3.

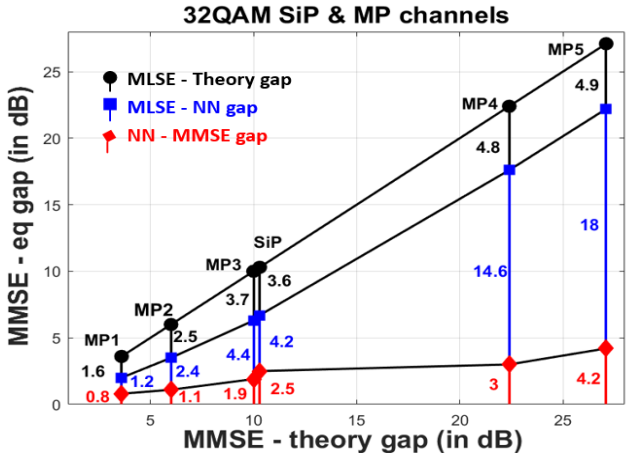
For all three modulations, we can see similarities in the behavior as ISI worsens. The milder ISI channels (low MMSE-theory gap), of course, see little difference in performance between




(a)




(b)




(c)

Figure 3.7 – BiLSTM performance gaps for SiP & MP channels for (a) 8QAM, (b) 16QAM, and (c) 32QAM (relative penalty in dB noted next to stem).

equalizers; even the linear equalizer performs well. As the ISI becomes more severe (moving right on the $x$-axis), our collection of parameterized channels manifests roughly linear growth in the performance gaps. We observe that the unrecoverable performance seems to saturate, so that at large $x$ the theory-MLSE gap (gray zone thickness) becomes roughly constant. In other words, for large $x$ the grey zone lower boundary tends to run parallel to the upper boundary, i.e., the $x = y$ line.

Consider now how NN performance deteriorates as we move to higher modulation order. Though not included, the QPSK plot would show zero gap between the NN and MLSE - the blue region would not be present. We see the blue region increase with increasing modulation order. Therefore our NN can no longer achieve optimal performance. However, the red region never disappears completely. The NN can always provide improvement over the linear solution.

For 16QAM the NN can recover roughly half the performance gap between the practical (MMSE) and the optimal (MLSE). The larger the combined blue/red regions, the more room for improvement our channel has in using a NN over the practical linear case. For channels with severe ISI, the NN can offer 4.2 dB of gain over a linear solution even for 32QAM.

## 3.7 Discussion

We examined the regression (MSE criteria) and classification versions (CE criteria) of all NNs. In Fig. 3.5(b) for QPSK we saw that the FFNN and LSTM achieved similar performance for CE; this was also true for MSE results. For higher order modulation formats the CE performance was best, so we use CE for all results presented in this paper.

### 3.7.1 FFNN vs. LSTM with sliding window inputs

As the LSTM and FFNN training achieved the same performance, we compared the weights in the hidden layer. For instance, at 11 dB SNR for MP5 channel, these two NN solutions had only 50% of their erroneous symbols in common. Therefore, they converged to different solutions, but with equal performance. Both FFNN and LSTM had sliding window inputs. We were surprised that despite the feedback available in the LSTM, it did not outperform the FFNN. We concluded that, while the LSTM led to a distinct solution, it was not exploiting the LSTM cells for sequence detection. Reducing the inputs to a single pair of IQ inputs did somewhat improve LSTM performance.

### 3.7.2 Convergence of BiLSTM

To examine the convergence of the CE error in classification for the deep BiLSTM, we examine two multipath channels: channel MP2 with moderate ISI and channel MP5 with high ISI. We consider 16QAM modulation with 25 dB SNR where the NN can achieve 1.2E-5 and 1.8E-2 BER for channels MP2 and MP5, respectively. The learning curves (not shown) for channel

MP2 are smooth with fast convergence with a few hundred epochs. Even for much lower SNR, the MP2 convergence was not problematic.

In Fig. 3.8 we present the CE learning curve for the validation set for deep BiLSTM. We see clear convergence anomalies, with spikes appearing often in the learning curve. The greater ISI of channel MP5 leads to less robust convergence. This is not unexpected as the ISI distortion makes training challenging. To overcome this behavior, we regularly saved the NN parameters. As seen in the red traces in Fig. 3.8(c), when the error increases the parameters are discarded. Once a sufficient number of epochs has been examined, we recover the parameters with the lowest error. We use this parameter set in the deep BiLSTM architecture to estimate the BER. Saving the parameter is important for high ISI and extremely low SNR, but is also beneficial for other scenarios.

### 3.7.3 Scalability of MLSE versus deep BiLSTM

For modulation order $M$ and a channel represented by $L$ taps in an FIR filter, i.e., channel memory of $L-1$ symbols, the MLSE equalizer computes $M^{L-1}$ metrics for each new received symbol. Due to this exponentially increasing complexity, the MLSE receiver is infeasible to implement for higher order QAM (16, etc.) and/or long channel memory (five symbols and more). Common hardware solutions for wireless communications can handle $2^9 = 256$ states. The MLSE is an excellent indicator of optimal performance, but is unattainable in higher order modulation systems.

The NN solutions we examined had complexity that we held constant with modulation order. Only the deep BiLSTM the output layer grows linearly with $M$; the majority of complexity remains constant. Our examination of 30 to 100 nodes and 2 or 3 hidden layers did not see significant performance improvement over the 60-node/2-layer solution. For QPSK we attained MLSE, but performance decreased as we moved to 32QAM. However 8QAM and 1QAM saw

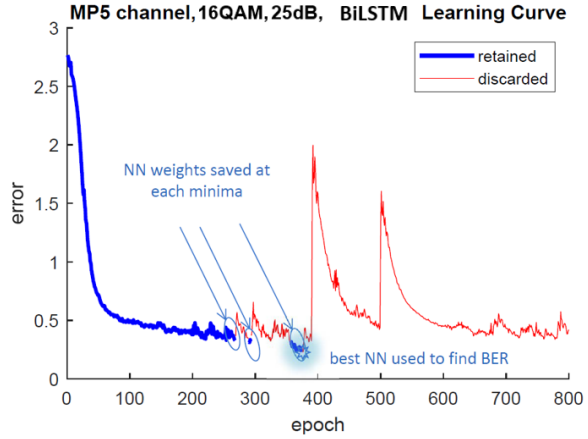
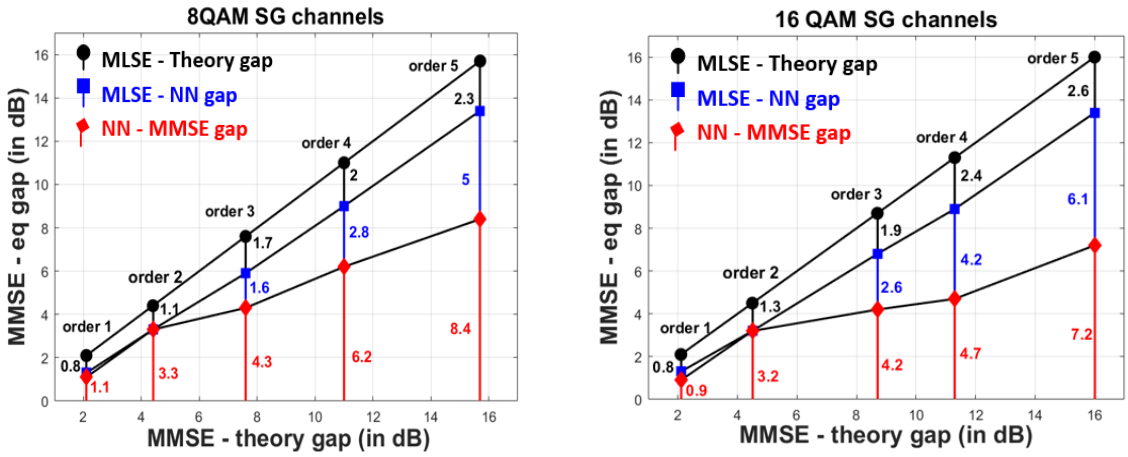



Figure 3.8 – Learning curves for MP5 channel using BiLSTM for 16QAM.




(a) (b)

Figure 3.9 – BiLSTM performance gaps for SG channels for (a) 8QAM, (b) 16QAM.

significant improvement in performance - and greatly outperformed the linear solution.

The memory length did not appear to offer significant impact on performance of the deep BiLSTM with fixed complexity. The super Gaussian family has memory length 4, while the memory length is 2 for the multipath channels we examined in section 3.6. In Fig. 3.9(a) and (b) we report super Gaussian results for 8QAM and 16QAM, respectively. At 32QAM ($M = 32$) transmission for SG channels ($L = 5$), our MATLAB simulator could not handle the $32^4$ states in the decoder trellis, hence we have no results for MLSE for this channel.

From Fig. 3.2 we can see that the multipath family has some frequency response shapes exhibiting dips at higher frequency. For the super Gaussian family, the roll-off is the primary change from one channel to another, with some shallow dips at low frequency. The $y$-axis scale has changed from Fig. 3.6 to Fig. 3.9 as the multipath family has more severe ISI. Nonetheless, despite these differences and the greater channel memory length, the qualitative behavior of the two families is quite similar.

Future work should examine whether a clear scaling law can be discerned for the deep BiLSTM solution. Would significant increases in the complexity (beyond the sweep we made) uncover architectures that continued to achieve MLSE performance?

## 3.8 Conclusion

We examined several NN architectures to find that which is most effective in mitigating severe ISI in bandlimited channels. We successfully demonstrated (via simulations) that our proposed deep BiLSTM achieves optimal MLSE performance for QPSK. This NN exploits memory cells within each node and two independent but identical NNs that treat the data in the forward and the backward directions before outputting the equalized data.

We also examined how NN performance scaled as we swept the severity of the ISI, the length of channel memory, and the modulation level. The severity of the ISI impacts the best attainable performance, while the other two factors determine the complexity of traditional MLSE to achieve the best attainable performance. Performance was qualitatively similar for the two memory lengths examined. While performance degraded with modulation order, improvement compared to simple linear MMSE filtering was still compelling, even at 32QAM. Of particular importance, these benchmarks for MMSE and MLSE assumed perfect channel state information, while the NN solution used only the training set.

# Conclusion

The increasing demand for global data traffic enabled by coherent optical communication systems, must be met with more flexible and scalable channel equalization techniques. As channel complexity increases for the future optical communication systems, the performance of conventional equalizers decreases. The self-learning algorithms such as NN techniques hold the potential to provide solutions for these future challenges. In this thesis, we have investigated various NN architectures for mitigating severe ISI caused by band-limited components in high speed optical communication systems. Mitigating ISI improves the reliability of high data rate transmissions and lead us towards the future communications.

To that end, we proposed a novel deep BiLSTM architecture that achieves the same BER performance as an optimal MLSE receiver for QPSK. This deep BiLSTM architecture relies entirely on the inherent LSTM cells for gaining insight into the sequence correlations. The bidirectional structure ensures that both past and future symbol observations are considered for equalizing the present symbol. Finally, the deep structure enhanced the sequential learning and made the deep BiLSTM architecture very effective in capturing the memory effects caused by the band-limited components.

The FFNN architecture, in contrast, was insufficient to capture the sequential dependencies using the framed inputs. On the other hand, despite being a feedback (RNN) architecture, the LSTM with a framed input was unable to exploit additional sequential insight from inherent feedback path. The LSTM and FFNN, while having very different architectures, converged to similar performance when trained with either CE or MSE criteria for QPSK. We realized that in the case of ISI mitigation, too much input context to the LSTM architecture inhibits its learning from the past.

Using deep BiLSTM architecture, we extended our examinations beyond QPSK for higher QAM transmission, where the ISI impact becomes much severe. This decreased data quality played a important role in the achievable NN performance. At higher QAM, the deep BiLSTM could only approach the optimal MLSE with a penalty; and also the increase in number of nodes or layers had no further impact on the attainable performance. We also demonstrated that, BiLSTM-MLSE penalty increases with the increase in modulation order, but it significantly outperforms the linear optimal MMSE solution, even at 32QAM. We examined the

performances for an experimental SiP channel and two families of ISI channels (one multipath family and another super Gaussian family) to sweep through the ISI severity. A new graphical approach to quantify the ISI impact based on baseline MLSE-MMSE performance gaps was presented. Using this approach, we illustrated how the deep BiLSTM performance decreases with the increase in ISI severity and modulation order.

The traditional MLSE complexity scales very poorly with QAM order and memory length, while the deep BiLSTM has a complexity which grows only linearly with the modulation order. In practical systems, due to the unavailability of perfect knowledge of the underlying channel, MLSE couldn't achieve optimal performance. The proposed deep BiLSTM does not require any explicit CSI to obtain the reported performances and it learns to equalize the channel in a data-driven fashion. Once the network is trained, it could be directly deployed for testing.

## Future Scope and Impacts

For the future research, there are different directions and aspects of this work that could be studied. In the following sections, we briefly review them.

### Channel with nonlinearity and ISI distortions

In this work, we developed our NN architectures with the focus to mitigate severe ISI which is a linear distortion effect, but this aspect of the research could also be extended to the other channels with both non-linearity and ISI distortion effects. Equalizing nonlinear ISI can be quite challenging and the MLSE implementation in a trellis structure is only applicable to a linear channel. The NN equalizer performs a complex mapping between input and output spaces. Decision regions can be involved with nonlinear decision boundaries. We believe that a NN can better model the nonlinear ISI and provide an efficient solution. In [65], authors showed that, when the nonlinear effect of the system is strong, the NNs can significantly outperform conventional algorithms. In channels with severe ISI and non-linearity, it would be interesting to analyze which of our NN exhibits improved performance vis-à-vis conventional receivers, and how the achievable performance varies with non-linearity.

### Transfer Learning

For the results reported in this work, we train and test the data at a fixed SNR. The NN performance decreases with the deviation in training and test data statistics, i.e., with a difference in SNR. This increases the training burden to train at each SNR, especially in changing SNR conditions. Transfer learning (TL) [66; 67] could be of great aid here. Using this deep learning technique, the knowledge acquired and preserved while learning about one task (source system), is transferred and used in the same way to solve related tasks (target system). Thus, TL-aided NNs can use a pre-trained model instead of learning from scratch.

This way of transferring knowledge from a different but related problem reduces the required numbers of training samples and convergence time. The more closely related the source and the target systems are, the more relevant information is transferred and the lower is the computational burden for training a new system.

In [68], in a 50-Gb/s 20-km PAM4 target system, TL based NN-equalization is performed using pre-trained NNs from a source systems of different bit rates or fiber lengths. The authors experimentally achieved 90% reduction in epochs and 56% in the number of training symbols using the pre-trained NNs from most similar source systems (50-Gb/s 25km), while attaining the same performance if trained from scratch. Inspired by these results, we believe that one can likely use TL aided NN equalization for coherent detection systems. This could greatly reduce the training time and computational burden while training a new target system in related SNR or related ISI impact (for e.g. MP2-MP3 and SG2-SG3) conditions. These techniques can be critical for fast and dynamic equalizers for future systems.

**Transformers**

Motivated by the recent success in LSTMs in fields of natural language processing (NLP) [57; 58; 59; 60; 61], we examined LSTM architectures for severe ISI mitigation and processing long-term sequences. However, at higher QAM our deep BiLSTM performance degraded with the data quality, i.e., not efficiently processing the sequences in presence of severe unwanted effects. Today, NLP is driven by the use of attention models like transformers [69], and are used more and more by Google [70; 69], Facebook [71] and Cortana. In context of NLP, transformers address the issues in LSTMs training, including poor performance in processing very long sequences and also concerns about processing times. Transformers solve these issues and were created as a combination of CNNs with attention models for language translation. In contrast to word by word processing as in LSTMs, CNNs used in transformers aid in the parallel processing of the words and translate the longer sequences faster.

Transformers embed each input word into a hidden state using an encoder, and the decoder takes into account every input word by considering its corresponding hidden state. Using attention architecture, it pays required attention to every input word and then processes the output, with the idea that there might be relevant information in every word in a sentence. We believe that these transformers can also efficiently process the sequences with poor data quality (severe ISI at higher QAM), as they do not depend on sequential learning path like LSTMs. Nevertheless, the internal architecture of transformers (encoder-decoders, attention models, word embedding) and the programming libraries are rigid to processing word sequences. Considerable effort will be needed to implement them in the context of ISI mitigation. However, this would be a great area to explore if libraries become flexible for numerical sequences. Transformers should be examined to determine if they can improve deep BiLSTM performance at higher QAM.

### Experimental Validation

This project could be to extended to verify the deep BiLSTM performance experimentally for electrical back-to-back transmission, and eventually for a SiP modulator. The proposed NN equalizers could also be effective for mitigating ISI caused by chromatic dispersion in the optical fibers and fading in wireless channels. Using NNs in practical applications, we could avoid the complexity of channel estimation process, and the receiver learns to self-optimize based on training data.

# Appendix A

# Appendix

In this appendix, we present the BER vs. SNR performance of deep BiLSTM vis-à vis conventional optimal equalizers, for QPSK transmission. In fact, we recreate the BER curves in Fig. 3.5 for all our remaining bandlimited channels. From the set of Figs. A.1 and Figs. A.2, it can be observed that, for all the MP and SG channels, the deep BiLSTM achieves the same performance of MLSE, i.e., zero penalty at 7% FEC threshold.

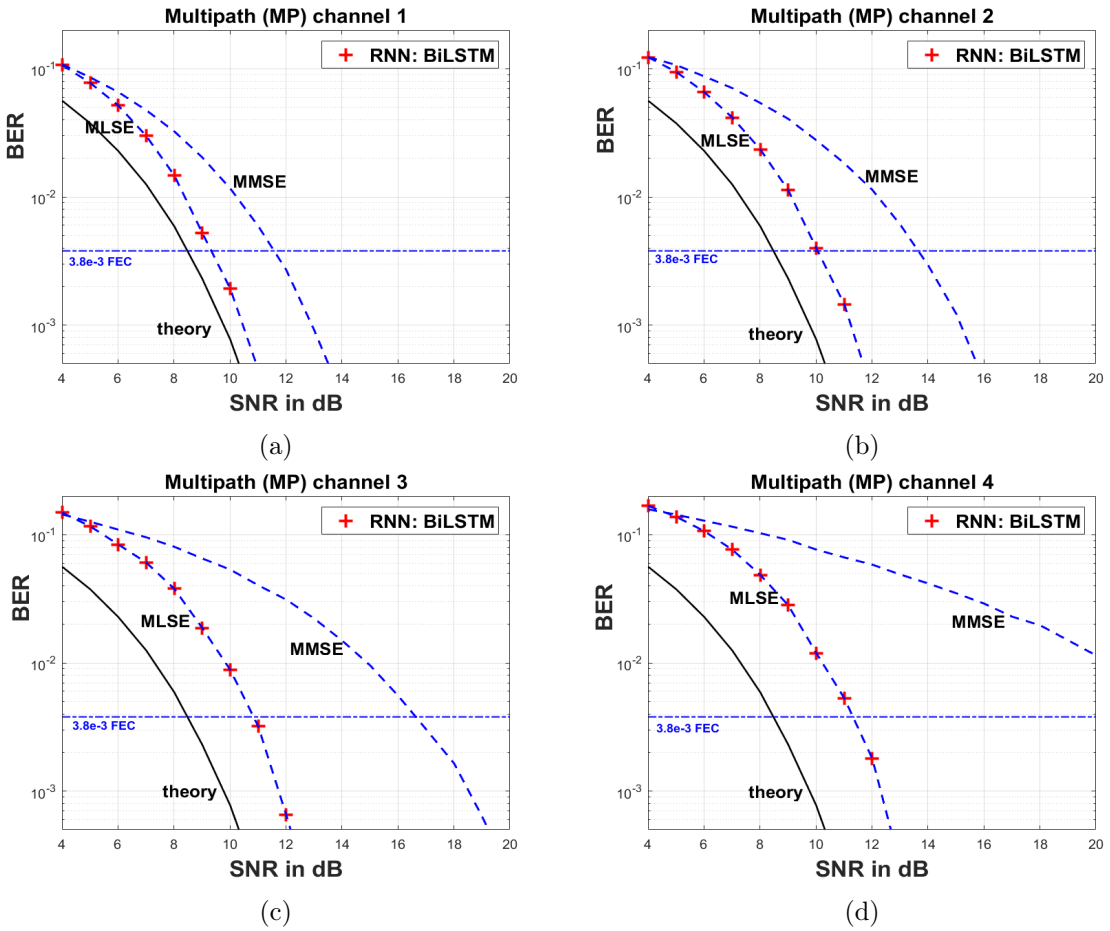



Figure A.1 - QPSK BER performance of BiLSTM for four Multipath (MP) channels.

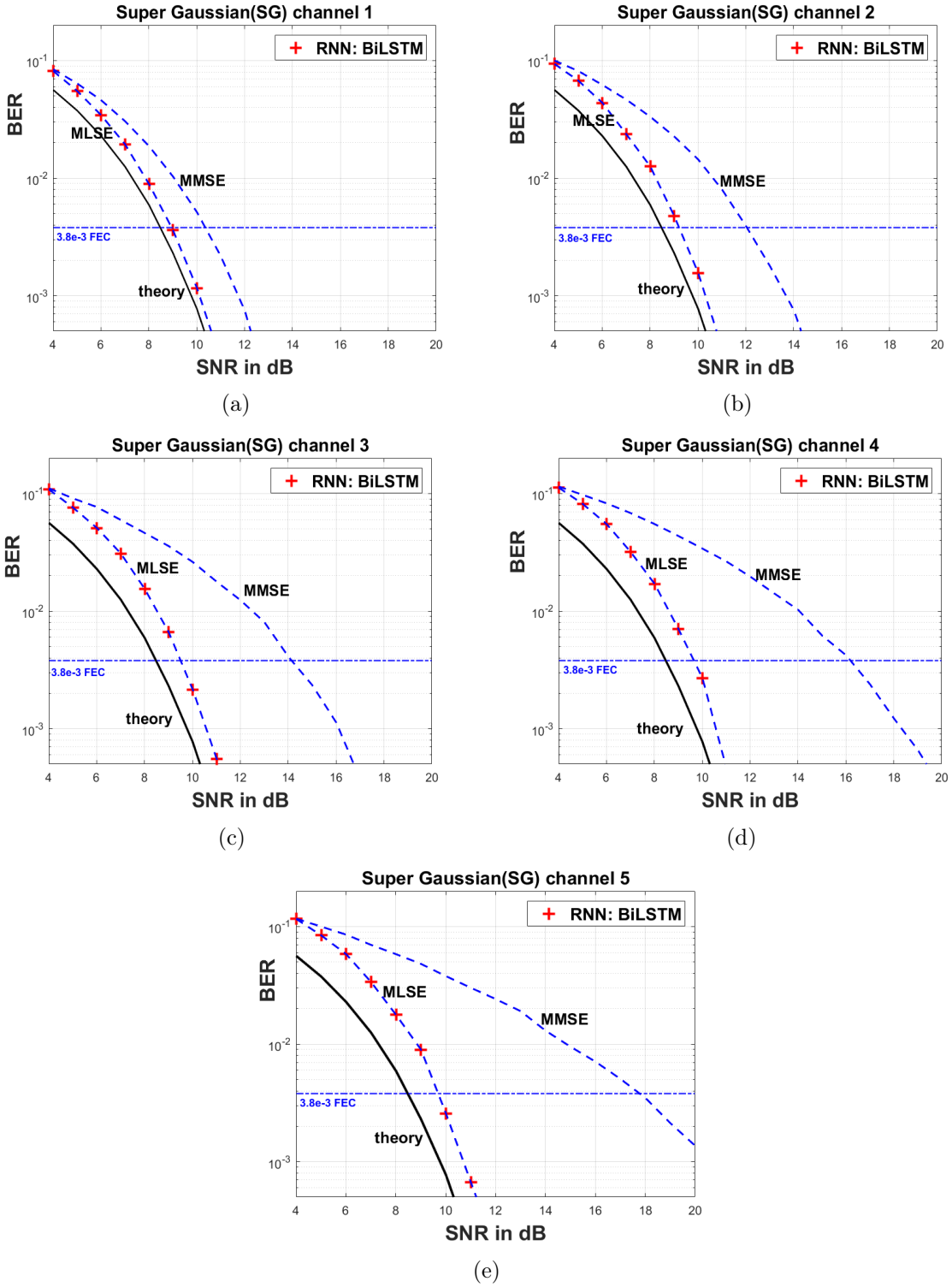



Figure A.2 – QPSK BER performance of BiLSTM for five Super-Gaussian (SG) channels.

# Appendix B

# Appendix

In this appendix, we give an overview of the programming files used in this work (all files are available online). We also provide the pseudo code for NN training. We simulated our entire project using two programming languages: Python and MATLAB. Python is used for our NN equalization, whereas MATLAB is mostly used for plotting frequency responses, calculating BER and for generating theoretical, MLSE, and MMSE BER curves.

We have five important MATLAB files used in this work, their functionalities are described as below:

1. *MLSE_ MMSE_ MQAM:* Calculates the BER values for MLSE, MMSE equalizers for M-QAM transmission.
2. *MMSE_ eq:* This implements the equations for training based MMSE receiver; this file is used to calculate MMSE equalizer output in the above file.
3. *BER_ calculation:* We save the input, output and target data of NN equalizers at each SNR as .mat files, and later use this "BER_calculation" file to calculate the input and output BER values of the NN.
4. *Freq_ response_ MP and SiP & Freq_ response_ SG:* These two files are used to plot the frequency responses of our examined channels in Chapter 3.
5. *SGC_ Taps:* This file calculates the five significant taps of SG channel at BW 150 GHz.

The codes implemented for the NN equalization (in Chapter 3) are divided into five major Python (.ipynb) files.

- *FFNN_ CE_ MSE_ QPSK:* Using this file, a FFNN can implemented for either CE or MSE criteria for QPSK.
- *LSTM_ CE_ MSE_ QPSK:* The file consists the implementation of four major blocks: LSTM using CE and MSE; deep BiLSTM using CE and MSE, for QPSK.

- *LSTM_ CE_ MSE_ 8QAM*, *LSTM_ CE_ MSE_ 16QAM* and *LSTM_ CE_ MSE_ 32QAM* implements the deep BiLSTM architecture for 8, 16 and 32QAM respectively.

The five Python files follow similar flow in their overall execution, with some minor differences regarding the QAM order and NN architecture. In the next part, we describe the pseudo code for deep BiLSTM using CE criteria at 16QAM. The flow diagram of the same pseudo code is also shown in Fig. B.1.

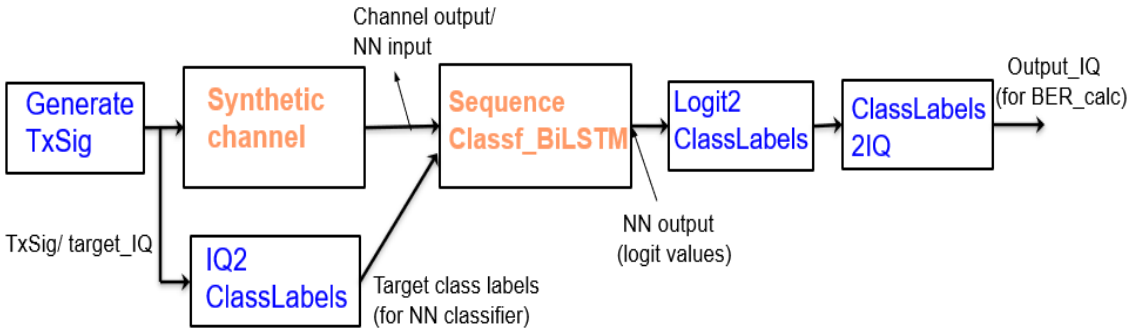



Figure B.1 – Flowchart for the pseudo code.

The following are the *user defined classes* and *user defined functions* used in our code:

- *Synthetic Channel* This class is synthetic channel model, it adds ISI memory effect and AWGN noise to the transmitted symbols.
- *SequenceClassf_ BiLSTM* This class creates a deep BiLSTM model as a classifier.
- *GenerateTxSig* : This function generates $N$ random M-QAM modulated symbols
- *IQ2ClassLabels*: This function converts IQ values of data to Gray coded class labels.
- *Logit2ClassLabels*: This function converts the Logit values of the deep BiLSTM output layer into most probable class labels.
- *ClassLabels2IQ*: This function converts Gray coded class labels back to IQ value.

**Pseudo code for NN equalization using deep BiLSTM classifier at 16QAM**

```
## Importing required functions and pytorch libaries
   import scipy.io as sio
   from matplotlib import pyplot as plt
   import numpy as np
   import torch
   import torch.nn as nn
   from torch.optim import SGD
   import torch.nn.functional as F
   import random
   from sklearn.metrics import mean_squared_error

## Main function
if __name__ == "__main__":
    ### Input parameters for Synthetic channel
    ch_memory = True                              ## Boolean value: whether to add memory effect or not
    ch_noise = True                               ## Boolean value: whether to add noise or not
    SNR_dB = 15.02
    h_channel = [0.2477+ 0.37668j ,0.7534+0.8400j ,0.4219+ 0.0893j]    ##SiP channel taps
    h_channel = h_channel/np.sqrt(np.sum((np.abs(h_channel))**2))     ##Normalizing channel energy to 1
    hI_channel = np.real(h_channel)
    hQ_channel = np.imag(h_channel)

### Creating Synthetic channel class with above inputs
    ch = SyntheticChannel(hI_channel,hQ_channel,ch_memory,ch_noise,SNR_dB)

## Defining NN parameters, optimizers and loss functions
    device = 'cuda'
    loss = nn.CrossEntropyLoss(reduction='mean')
    optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
    channel_taps = len(h_channel)
    trainerror = [ ]
    valerror= [ ]
    PATH = 'model_BiLSTM_16QAM'        ##PATH to save NN model of minimum error

### Creating deep BiLSTM Classifier
    model = SequenceClassf_BiLSTM()      ##network configuration is defined inside the class
    model.to(device)

### Generating random validation data and sending to synthetic channel
    Valch_input = torch.as_tensor(np.array(generateTxSig(10000)),dtype = torch.float)
    Rx_X_val = ch.forward(Valch_input)       ## output of the channel
    if ch_memory == False:
        Valch_input = Valch_input                ## channel input or target data
    else:
        Valch_input = Valch_input[memory_duration-2:-(memory_duration-2)]
    val_input = torch.as_tensor(np.array(Rx_X_val),dtype = torch.float)
```

```
### DBiLSTM training and monitoring performance of train, validation sets ########################
  for n in range(850):            ## one loop is one epoch training
    ## Generating random training data and sending to synthetic channel
    Trainch_input = torch.as_tensor(np.array(generateTxSig(10000)),dtype = torch.float)
    Rx_X_train = ch.forward(Trainch_input)
    if ch_memory == False:
       Trainch_input = Trainch_input
    else:
       Trainch_input = Trainch_input[memory_duration-2 :-(memory_duration-2)]
    train_input = torch.as_tensor(np.array(Rx_X_train),dtype = torch.float)
    target_IQ = Trainch_input                  ## target IQ (or) channel input IQ
    target_IQ =  target_IQ[1:-1]
    target = IQ2ClassLabels(target_IQ*3.1622)   ## ClassLabels as targets to train the classifier
    target = target.to(device)
    train_input = train_input.to(device)

    ## NN weight updates using CE criteria and Adam optimizer
    optimizer.zero_grad()
    output = model(train_input)                ## output of the NN
    a = output[:,0,0].cpu().detach().numpy().size
    output = output.view(a,16)
    CE_train= loss(output,target)
    train_error = CE_train.cpu().detach().numpy()
    trainerror.append(train_error)
    CE_train.backward(retain_graph=True)    ## gradient calculation by back propogation
    optimizer.step()                                ## NN weights update

    ### Checking the model performance on validation set
    val_target_IQ = Valch_input
    val_target_IQ = val_target_IQ[1:-1]
    val_target = IQ2ClassLabels(val_target_IQ*3.1622)
    val_target = val_target.to(device)
    val_input = val_input.to(device)
    val_output = model(val_input)
    b = val_output[:,0,0].cpu().detach().numpy().size
    val_output = val_output.view(b,16)
    CE_val = loss(val_output, val_target)                    ##calculating validation CE error
    val_error = CE_val.cpu().detach().numpy()
    valerror.append(val_error)
    print(train_error,val_error,n)

    ## Saving and Updating NN model parameters corresponding to minimum validation error.
    if n > 100:
      if val_error == np.min(valerror):
        torch.save(model.state_dict(),PATH)
        print("new weights model saved with least error", val_error)
```

```
###### Plotting learning curves ############
    plt.figure()
    plt.plot(trainerror,label ="Train error")
    plt.plot(valerror, label ="validation error")
    plt.title("CE error vs no of epochs")

### Loading NN model corresponding to minimum error occured in training ########################
    model2 = SequenceClassf_BiLSTM()                              ##create dummy class model
    model2.load_state_dict(torch.load('model_BiLSTM_16QAM'))  ##loading trained NN parameters
    model2.to(device)

### Generating random test data and sending to synthetic channel
    Testch_input = torch.as_tensor(np.array(generateTxSig(100000)),dtype = torch.float)
    Rx_X_Test = ch.forward(Testch_input)
    if ch_memory == False:
        Testch_input = Testch_input
    else:
        Testch_input = Testch_input[memory_duration-2:-(memory_duration-2)]
    Test_input = torch.as_tensor(np.array(Rx_X_Test),dtype = torch.float)

### Preparing test data and sending to NN model
    Test_target_IQ = Testch_input
    Test_target_IQ = Test_target_IQ[1:-1]
    Test_target = IQ2ClassLabels(Test_target_IQ*3.1622)
    Test_target = Test_target.to(device)
    Test_input = Test_input.to(device)
    Test_output = model2(Test_input)                        ##test output
    c = Test_output[:,0,0].cpu().detach().numpy().size
    Test_output = Test_output.view(c,16)
    CE_test = loss(Test_output,Test_target)              ## calculating CE error of test data
    Test_error = CE_test.cpu().detach().numpy()
    print("Test error value with min error saved model",Test_error.mean())

##### Saving input, output and target of test data as .mat files for BER_calculation #########
    Test_ClassLabels = Logit2ClassLabels(Test_output)          ##  NN output (logit values) to classlabels
    Testpredicted_IQoutput = ClassLables2IQ(Test_ClassLabels)   ## getting output IQ  for BER_calc
    origin = np.zeros_like(Test_target_IQ)
    Test_target_IQ = Test_target_IQ.cpu().detach().numpy()
    Test_input = Test_input.cpu().detach().numpy()
    Ps_target = mean_squared_error(Test_target_IQ ,origin)*2
    Ps_input = mean_squared_error(Test_input[1:-1,:2] ,origin)*2
    tr = 3.1622/np.sqrt(Ps_target)                                ## for re-normalizng the energy to IQ space
    ip= 3.1622/np.sqrt(Ps_input)
    sio.savemat('Equalized_output.mat', {'test_output':Testpredicted_IQoutput})
    sio.savemat('targetData.mat', {'test_target':tr*Test_target_IQ})
    sio.savemat('MemoryNoisy_input.mat', {'test_input':ip*Test_input[1:-1]})
```

# Publication list

## Published papers

1. Sai Chandra Kumari Kalla, Rizan Homayoun Nejad, Sasan Zhalehpour, and Leslie Ann Rusch. "Neural Nets to Approach Optimal Receivers for High Speed Optical Communication," In *CLEO: Science and Innovations*, pp. STh4M-4, Optical Society of America, 2020
2. Sai Chandra Kumari Kalla, and Leslie Ann Rusch "Recurrent neural nets achieving MLSE performance in bandlimited optical channels," In *IPC: Machine Learning in Photonics Systems III*, pp. MA3-3, 2020

## Submitted papers

1. Sai-Chandra-Kumari Kalla, Christian Gagné, and Leslie A. Rusch, "Recurrent Neural Networks Achieving MLSE Performance for Optical Channel Equalization," submitted, *Journal of Light-wave Technology (JLT)*, October 2020.

# Bibliography


[1] S. Spolitis and G. Ivanovs, "Realization of combined chromatic dispersion compensation methods in high speed WDM optical transmission systems," *Elektronika ir Elektrotechnika*, vol. 113, no. 7, pp. 101–106, 2011.

[2] S. C. K. Kalla, C. Gagné, and L. A. Rusch, "Recurrent Neural Networks Achieving MLSE Performance for Optical Channel Equalization," submitted*, Journal of Light-wave Technology (JLT)*, October 2020.

[3] S. C. K. Kalla and L. A. Rusch, "Recurrent neural nets achieving MLSE performance in bandlimited optical channels," in *2020 IEEE Photonics Conference (IPC)*, pp. MA3–3, IEEE, 2020.

[4] S. C. K. Kalla, R. H. Nejad, S. Zhalehpour, and L. A. Rusch, "Neural Nets to Approach Optimal Receivers for High Speed Optical Communication," in *CLEO: Science and Innovations*, pp. STh4M–4, Optical Society of America, 2020.

[5] S. Zhalehpour, M. Guo, J. Lin, Z. Zhang, Y. Qiao, W. Shi, and L. A. Rusch, "System Optimization of an All-Silicon IQ Modulator: Achieving 100-Gbaud Dual-Polarization 32QAM," *Journal of Lightwave Technology*, vol. 38, no. 2, pp. 256–264, 2019.

[6] M. Castells, *The information age*, vol. 98. Oxford Blackwell Publishers, 1996.

[7] C. V. Networking, "Cisco global cloud index: Forecast and methodology, 2015-2020. white paper," *Cisco Public, San Jose*, 2016.

[8] G. Keiser, "Optical fiber communications," *Wiley encyclopedia of telecommunications*, 2003.

[9] P. J. Winzer, D. T. Neilson, and A. R. Chraplyvy, "Fiber-optic transmission and networking: the previous 20 and the next 20 years," *Optics express*, vol. 26, no. 18, pp. 24190–24239, 2018.

[10] E. Ip, A. P. T. Lau, D. J. Barros, and J. M. Kahn, "Coherent detection in optical fiber systems," *Optics express*, vol. 16, no. 2, pp. 753–791, 2008.

[11] K. Kikuchi, "Fundamentals of coherent optical fiber communications," *Journal of Lightwave Technology*, vol. 34, no. 1, pp. 157–179, 2015.

[12] S. J. Savory, "Digital coherent optical receivers: Algorithms and subsystems," *IEEE Journal of selected topics in quantum electronics*, vol. 16, no. 5, pp. 1164–1179, 2010.

[13] Cisco, "Cisco visual networking index: Forecast and methodology, 2016– 2021," *CISCO White paper*, 2017.

[14] J. G. Proakis, *Digital Communications*. McGraw-Hill Education, 2000, Chap. 10.

[15] A. Klein, G. K. Kaleh, and P. W. Baier, "Zero forcing and minimum mean-square-error equalization for multiuser detection in code-division multiple-access channels," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 2, pp. 276–287, 1996.

[16] B. Sklar, "How I learned to love the trellis," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 87–102, 2003.

[17] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, Nov. 2016.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[20] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[21] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, 2018.

[22] M. Ibnkahla, "Applications of neural networks to digital communications–a survey," *Signal processing*, vol. 80, no. 7, pp. 1185–1215, 2000.

[23] S. Dörner, S. Cammerer, J. Hoydis, and S. Ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2017.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] S. Luyi, F. Jinyi, and Y. Xiaohua, "Forward error correction," in *2012 Fourth International Conference on Computational and Information Sciences*, pp. 37–40, 2012.

[26] Ting Chen, "Analysis of forward error correcting codes," in *2011 International Conference on System science, Engineering design and Manufacturing informatization*, vol. 1, pp. 329–332, 2011.

[27] G. Tzimpragos, C. Kachris, I. B. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos, "A survey on FEC codes for 100 G and beyond optical networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 209–221, 2014.

[28] R. Schmogrow, M. Winter, M. Meyer, D. Hillerkuss, B. Nebendahl, J. Meyer, M. Dreschmann, M. Huebner, J. Becker, C. Koos, *et al.*, "Real-time Nyquist pulse modulation transmitter generating rectangular shaped spectra of 112 Gbit/s 16QAM signals," in *Signal Processing in Photonic Communications*, p. SPMA5, Optical Society of America, 2011.

[29] B. Sklar *et al.*, *Digital communications: fundamentals and applications*. 2001.

[30] J. McCarthy and E. A. Feigenbaum, "In memoriam: Arthur Samuel: Pioneer in machine learning," *AI Magazine*, vol. 11, no. 3, pp. 10–10, 1990.

[31] R. O. Duda, P. E. Hart, *et al.*, *Pattern classification and scene analysis*, vol. 3. Wiley New York, 1973.

[32] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, "Backpropagation: The basic theory," *Backpropagation: Theory, architectures and applications*, pp. 1–34, 1995.

[33] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[34] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[35] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[36] H. Robbins and S. Monro, "A stochastic approximation method In: Herbert Robbins Selected Papers," *NewYork, USA: Springer*, vol. 102, p. 109, 1985.

[37] J. L. McClelland, D. E. Rumelhart, P. R. Group, *et al.*, "Parallel distributed processing," *Explorations in the Microstructure of Cognition*, vol. 2, pp. 216–271, 1986.

[38] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[40] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5595–5637, 2017.

[41] M. Bartholomew-Biggs, S. Brown, B. Christianson, and L. Dixon, "Automatic differentiation of algorithms," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 171–190, 2000.

[42] J. Makka, H. Monga, and S. Baghla, "Reduction of Inter-Symbol Interference Using Artifical Neural Network System in Multicarrier OFDM System," *Electric Electron Tech Open Acc J*, vol. 2, no. 3, pp. 94–97, 2018.

[43] F. Bouguerra, I. Benacer, and L. Saidi, "MLP and RBF symbol tracking with 16 QAM modulation over multipath distorted channel," in *2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, pp. 182–187, IEEE, 2017.

[44] C.-Y. Lo *et al.*, "Application of neural network techniques on nonlinear channel equalization for 16-QAM modulation systems," in *2008 Eighth International Conference on Intelligent Systems Design and Applications*, vol. 1, pp. 356–361, IEEE, 2008.

[45] K. Hacioglu, "An improved recurrent neural network for M-PAM symbol detection," *IEEE Transactions on neural networks*, vol. 8, no. 3, pp. 779–783, 1997.

[46] M. A. Jarajreh, E. Giacoumidis, I. Aldaya, S. T. Le, A. Tsokanos, Z. Ghassemlooy, and N. J. Doran, "Artificial neural network nonlinear equalizer for coherent optical OFDM," *IEEE Photonics Technology Letters*, vol. 27, no. 4, pp. 387–390, 2014.

[47] J. C. Patra, R. N. Pal, R. Baliarsingh, and G. Panda, "Nonlinear channel equalization for QAM signal constellation using artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 2, pp. 262–271, 1999.

[48] H. C. Myburgh and J. C. Olivier, "Near-optimal low complexity MLSE equalization," in *2008 IEEE Wireless Communications and Networking Conference*, pp. 226–230, IEEE, 2008.

[49] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: A deep learning based Viterbi algorithm for symbol detection," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3319–3331, 2020.

[50] Y. Hsu, C.-Y. Chuang, Y. Tong, C.-W. Chow, J. Chen, Y.-C. Lai, C.-H. Yeh, Y.-K. Chen, and H. K. Tsang, "Implementing Deep Neural Network for Signal Transmission Distortion Mitigation of PAM-4 Generated by Silicon Mach-Zehnder Modulator," in *in 2019 24th OptoElectronics and Communications Conference (OECC)*, pp. 1–3, IEEE, 2019.

[51] N. Farsad and A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Transactions on Signal Processing*, vol. 66, no. 21, pp. 5663–5678, 2018.

[52] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[54] N. Ryant, M. Liberman, and J. Yuan, "Speech activity detection on YouTube using deep neural networks," in *INTERSPEECH*, pp. 728–731, Lyon, France, 2013.

[55] H. Zhao, S. Zarar, I. Tashev, and C.-H. Lee, "Convolutional-recurrent neural networks for speech enhancement," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2401–2405, IEEE, 2018.

[56] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE Access*, vol. 6, pp. 1155–1166, 2017.

[57] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4520–4524, IEEE, 2015.

[58] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," in *in Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 338–342, 2014.

[59] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International conference on machine learning (ICML)*, pp. 1764–1772, 2014.

[60] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[61] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

[62] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[63] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[64] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction," in *Proc. 6th Int. Workshop Urban Computing (UrbComp)*, 2016.

[65] L. Yi, T. Liao, L. Huang, L. Xue, P. Li, and W. Hu, "Machine learning for 100 Gb/s/$\lambda$ passive optical network," *Journal of Lightwave Technology*, vol. 37, no. 6, pp. 1621–1630, 2019.

[66] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[67] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264, IGI Global, 2010.

[68] Z. Xu, C. Sun, T. Ji, H. Ji, and W. Shieh, "Transfer Learning Aided Neural Networks for Nonlinear Equalization in Short-Reach Direct Detection Systems," in *2020 Optical Fiber Communications Conference and Exhibition (OFC)*, pp. 1–3, IEEE, 2020.

[69] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[70] "Google AI Blog Transformer: A Novel Neural Network Architecture for Language Understanding." https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html. Accessed: 2020-10-07.

[71] "Facebook engineering: A novel approach to neural machine translation." https://engineering.fb.com/ml-applications/a-novel-approach-to-neural-machine-translation/. Accessed: 2020-10-07.