Learning-Based Communication Systems

A Thesis

Presented in Partial Fulfillment of the Requirements for the

Degree of Master of Science

with a

Major in Electrical Engineering

in the

College of Engineering

University of Idaho

by

Wael Fatnassi

Major Professor: Zouheir Rezki, Ph.D.

Committee Members: James F. Frenzel, Ph.D.; Somantika Datta, Ph.D.

Department Administrator: Joseph Law, Ph.D.

May 2019

Authorization to Submit Thesis

This thesis of Wael Fatnassi, submitted for the degree of Master of Science with a major in Electrical Engineering and titled "Learning-Based Communications Systems", has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor:	______________________ Zouheir Rezki, Ph.D.	Date__________
Committee Members:	______________________ James F. Frenzel, Ph.D.	Date__________
	______________________ Somantika Datta, Ph.D.	Date__________
Department Administrator:	______________________ Joseph Law, Ph.D.	Date__________

Abstract

Connecting people offers opportunities to build communities of any size and consequently, brings the world closer together. Conventionally, the connectivity has happened through traditional radio-frequency communication method. However, the ever-increasing demand for higher data-rate communications and the explosion of advanced wireless applications such as virtual reality, augmented reality, and internet of things, reduces the effectiveness of these methods. Therefore, developing next-generation technologies, such as learning-based communication systems, that can satisfy the large data and ultra-high rate communication requirements would be of interest.

To address the challenging problem of connectivity, our research focuses on developing a learning-based framework for the next-generation communication systems. These systems can proactively adapt their communication and networking strategies to the dynamics of the environment, thereby maximizing their end-to-end performance in terms of data-rates, energy-efficiency, and link-reliability. Toward this goal, first information-theoretical tools are used to establish the fundamental limits (including bounds on the end-to-end performance). These performance limits are the keys for building reliable and efficient systems. Then, powerful machine learning techniques, such as deep learning, are employed for the implementation of such systems. In particular, a simple and cost-effective system with near-optimal performance can be implemented by merely taking off-the-shelf deep learning models, applying them to communication design problems, and tuning them based on the training data.

Vita

Education

- Master of Science in Electrical Engineering, University of Idaho, Idaho, USA, August 2017- May 2019.
- Bachelor in Telecommunication Engineering, Higher School of Communications, Tunis, Tunisia, September 2013-June 2016.

List of Publications

[J1] **W. Fatnassi** and Z. Rezki, "Reliability Enhancement of Smart Metering System Using Millimeter Wave Technology,"*IEEE Transactions on Communications*, vol. 66, no. 10, pp. 4877-4892, Oct 2018.

[J2] **W. Fatnassi** and Z. Rezki, "Training Deep Neural Networks for Partial Interference Cancellation in Uplink Cellular Networks," **Submitted** to *IEEE Wireless Communications Letters*, 2019.

[J3] A. Aboutaleb, **W. Fatnassi**, A. Chaaban, and Z. Rezki, "Error Performance of Space-Time Codes over MIMO Nakagami Fading Channels: Design Criteria, New Insights, and the Impact of Blockage," **Submitted** to *IEEE Transactions on Communications*, 2019 **[**2^{nd} **round review]**.

[C1] **W. Fatnassi** and Z. Rezki, "Increasing the Reliability of Smart Metering System Using Millimeter Wave Technology," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2018, pp.1-6.

[C2] **W. Fatnassi** and Z. Rezki, "Upper Bound on the Expected Generalization Error of the Convolutional Neural Network," **Submitted** to *International Symposium on Information Theory (ISIT)*, 2019.

[C3] A. Abutaleb, **W. Fatnassi**, M. Soltani, and Z. Rezki, “Symbol Detection and Channel Estimation Using Neural Networks in Optical Wireless Communications Systems”, *2019 IEEE International Conference on Communications (ICC 2019)* **[Accepted]**.

[C4] A. Aboutaleb, **W. Fatnassi**, A. Chaaban, and Z. Rezki, “On the Error Performance of Space-Time Codes over MIMO Nakagami Fading Channels with Blockage," in *Proceedings the 29th IEEE Biennial Symposium on Communications (BSC 2018)*, Toronto, ON, Canada, 6-7 June 2018.

[C5] M. Soltani, **W. Fatnassi**, A. Abutaleb, Z. Rezki, “Autoencoder-Based Optical Wireless Communications Systems”, *IEEE Global Conference on Communications Workshop on Machine Learning for Communications (MLCOMM'2018)*, Abu Dhabi, UAE, 9-13 December 2018 **[Accepted]**.

Academic Awards

GPSA TEGA Outstanding Graduate Student Award, University of Idaho, April 2019.

Work experiences

- **Internship at the University of Idaho from Nov. 2016 to Jun. 2017**: Modeled Multiple Access Channel at millimeter wave frequencies, and analyzed the path loss, the fading, the shadowing, and co-channel interference effects on the reliability of Smart Metering System. (Supervisor: Zouheir Rezki)

- **Internship at the Idaho National Laboratory (INL) in Summer 2018**: Surveying 5G enabling technologies, 5G testbeds and prototypes, such as POWDER at the University of Utah and COSMOS at WINLAB. (Supervisors: Arupjyoti Bhuyan, and Paul E. Titus)

Acknowledgements

I would like to thank my advisor Dr. Zouheir for his close supervision of this work and of my studies. Without his guidance, this research would not have been possible. While working with him, I was very impressed by his rigorous and comprehensive approach to research problems. I learned from him to be a better researcher. For that, I am indebted to him.

I would also like to thank my committee members, Dr. Frenzel and Dr. Datta, for kindly volunteering their time and efforts to improve my thesis.

I would also like to thank the ECE Department Administer, Dr. Joseph Law, for his continued help with all of the logistics involved with courses, contracts, and the submission of this thesis.

I would also like to thank the friends I made during my time at the University of Idaho: Elyes, Yassine, Mustafa, Ahmed, Morteza, Mohannad, Amalia, Cathy, Taha, and Syrine. Their companionship made this journey more enjoyable.

At last but not least, I would like to thank my family back in Italy, especially my mother for her encouragement and support through out my life and in particular during the last two years. Without her sacrifices, I would never have been able to reach this stage in life.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction and Background

1.1 Communication Systems

Communication systems transmit information from a source to a destination via a physical communication channel that propagates electromagnetic, acoustic, or optical waves. Figure 1.1 shows a block diagram of a communication system.

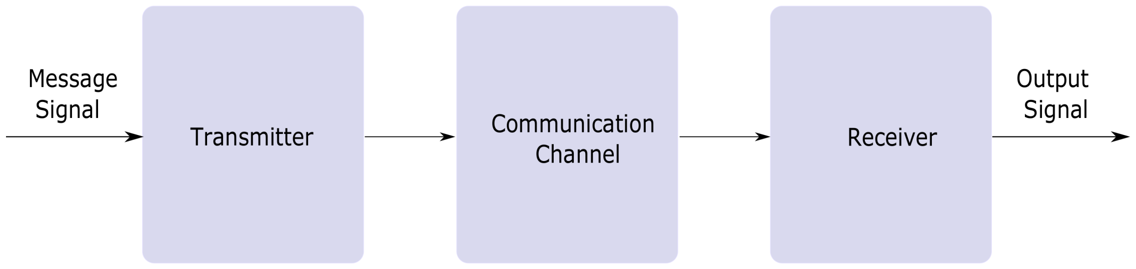


Figure 1.1: A block diagram of a communication system.

As shown in Fig. 1.1, a communication system has three main elements: A transmitter, a communication channel, and a receiver. The main purpose of the transmitter is to process the message signal into a form suitable for transmission over the physical communication channel. This process is called modulation. The communication channel's function is to provide a path between the transmitter's output and the receiver's input. The receiver's function is to recover the message signal from the received signal using detection and decoding algorithms. The transmitter can use two methods to transmit the message signal over the communication channel: Analog or digital methods. Both methods have advantages and disadvantages. Digital methods have the following advantages:

- They have an immunity to noise and interference which are impossible to prevent in the communication channel.
- The transmitter can implement the same format for different types of message signals including voice data or video data.

- The transmitter can use encryption techniques to provide security for message signals.

The disadvantages of the digital methods are that they can be complex and costly for communication channels such as underwater communication channels, satellite channels, and optical fibers. For analog methods, the advantages include the following:

- The implementation of most analog methods in communication systems is very simple.
- Analog methods are not expensive because they can be implemented using simple technologies.

1.2 Modulation/Demodulation

Modulation is the process that modifies the message signal into a form that is suitable for transmission over the communication channel. It changes some parameters of the carrier wave in accordance with the message signal so that the output signal will match the bandwidth of the communication channel. The modulation process happens on the transmitter side. To recover the message signal from the received signal, the receiver uses the inverse of the modulation process which is called the demodulation process. There are other benefits of using modulation other than matching the communication channel's bandwidth. One of these benefits is that modulation enables multiplexing. This process enables the transmitter to transmit different message signals over the same channel at the same time. The other benefit is that modulation modifies the message signal to be immune to noise and interference. The modulation process varies one or more parameters of the carrier signal, with a modulating signal that typically contains information of the message signal. The carrier signal is a sinusoidal signal and has three independent parameters that can be varied with the message signal. These three parameters are amplitude, frequency, and phase. The process that changes the carrier signal's amplitude in accordance with the message signal is called amplitude modulation (AM). Frequency modulation (FM) is the process that varies the frequency of the carrier wave. The last form of modulation is phase modulation which is done by changing the carrier signal's phase. Most of the communication systems use those modulations in our daily life.

1.3 Detection/Estimation Algorithms

The receiver uses algorithms to estimate the transmitted symbols from the received signal. In this section, we present detection algorithms used by the receiver to recover the information transmitted by the transmitter. Before we present those algorithms, we need to describe our system model so that we can have some insights for each algorithm's performance. We assume that we have a single-input-single-output (SISO) communication system. In other words, we have one transmitter/receiver and each one is equipped with one single transmit/receive antenna. The transmitter wishes to transmit p symbols $\boldsymbol{x} = [x_1, \cdots, x_p]$ to the receiver, over p time slots. We denote by $\boldsymbol{y} = [y_1, \cdots, y_p]$ the received symbols. The received signal $\boldsymbol{y}$ is related to the transmitted signal $\boldsymbol{x}$ through a probabilistic channel law $\mathcal{P}_{\boldsymbol{y}|\boldsymbol{x}}$ which is the probability law of observing $\boldsymbol{y}$ given that an $\boldsymbol{x}$ has been transmitted.

1.3.1 MAP Detection Algorithm

To find the maximum a-posteriori (MAP) estimate of $\boldsymbol{x}\,(i)$ given that we have observed $\boldsymbol{y}\,(i)$, we find the value of $\boldsymbol{x}\,(j)\,,\ 1 \leq j \leq p$, that maximizes

$$\hat{\boldsymbol{x}}\,(i) = \underset{\boldsymbol{x}(j),\ j=1,\ldots,p}{\arg\max}\ \mathcal{P}(\boldsymbol{y}\,(i)\,|\boldsymbol{x}\,(j))\mathcal{P}(\boldsymbol{x}\,(j)) \quad \text{for } 1 \leq i \leq p. \tag{1.1}$$

1.3.2 ML Detection Algorithm

The maximum likelihood (ML) detector searches for all the possible transmitted symbols $\boldsymbol{x}\,(j)\,,\ 1 \leq j \leq p$, to find the estimated symbol $\hat{\boldsymbol{x}}\,(i)$ that maximizes the likelihood of obtaining the received sequence [1], i.e.,

$$\hat{\boldsymbol{x}}\,(i) = \underset{\boldsymbol{x}(j),\ j=1,\ldots,p}{\arg\max}\ \mathcal{P}(\boldsymbol{y}\,(i)\,|\boldsymbol{x}\,(j)) \quad \text{for } 1 \leq i \leq p. \tag{1.2}$$

When the prior probabilities are uniform, i.e., $\mathcal{P}(\boldsymbol{x}\,(j)) = \frac{1}{p},\ \forall j \in \{1, \cdots, p\}$, the MAP estimator is equivalent to the ML estimator.

Although the above algorithms are optimal from an error performance point-of-view, they are complex to implement, especially for multiple-input-multiple-output (MIMO) systems. Next, we briefly overview the two most popular linear algorithms that are less complex, but are sub-optimal.

1.3.3 ZF Detection Algorithm

Zero forcing (ZF) algorithm is implemented in MIMO communication system. In this case, the transmitter has N_t transmit antennas while the receiver has N_r receive antennas. The transmitter wishes to transmit $N_t \times p$ symbols through N_t transmit antennas. We denote by $\boldsymbol{X}$, the $N_t \times p$ transmitted matrix. We denote by $\boldsymbol{Y}$, the $N_r \times p$ matrix that contains all the received signals. We consider the elements of the received matrix $\boldsymbol{Y}$ are given by the following equation:

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{X} + \boldsymbol{Z}, \tag{1.3}$$

where $\boldsymbol{H}$ is the $N_r \times N_t$ channel matrix that contains the communication channel's coefficients, and $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_{N_r})$ is a circularly additive white Gaussian noise (AWGN) with $\boldsymbol{0}$ mean and covariance $\sigma^2 \boldsymbol{I}_{N_r}$. We assume a flat fading environment, where the channel matrix $\boldsymbol{H}$ is constant over the transmission of p symbols. Examination of equation (1.3) shows that each received signal at each receive antenna consists of linear superposition of signals from all the transmit antennas. As a result, if a receive antenna tries to detect the signal from a specific transmit antenna, then the signals from the other transmit antennas constitute the interference. In a ZF detector, the effect of interference is reduced by premultiplying the received signal matrix $\boldsymbol{Y}$ by the Moore-Penrose pseudo inverse of the channel matrix, which is denoted by $\boldsymbol{H}^{+}$ and is defined as follows:

$$\boldsymbol{H}^{+} = \left(\boldsymbol{H}^{H}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^{H}, \tag{1.4}$$

where $\boldsymbol{H}^{H}$ is the conjugate transpose of the matrix $\boldsymbol{H}$.

1.3.4 LMMSE Detection Algorithm

In linear minimum mean square (LMMSE) detection algorithm, the estimated transmitted matrix $\hat{\boldsymbol{X}}$ is given by:

$$\hat{\boldsymbol{X}} = \boldsymbol{W}_{mmse}\boldsymbol{Y}, \tag{1.5}$$

where $\boldsymbol{W}_{mmse} = \left(\boldsymbol{H}^{H}\boldsymbol{H} + \sigma^{2}\boldsymbol{I}_{N_r}\right)^{-1}\boldsymbol{H}^{H}$. If we compare (1.5) and (1.4), we find that the difference between the two equations is the term $\sigma^{2}\boldsymbol{I}_{N_r}$. This term avoids the problem of noise amplification in ZF detection algorithm.

1.4 Thesis Outline

This thesis aims to provide efficient and practical learning-based algorithms for the communication system. First, we present the background and the important basis of deep learning (DL). Second, we propose information-theoretical tools to establish the fundamental limits and to understand the inner mechanism of DL algorithms. Finally, we apply DL techniques to the communication system for interference cancellation, channel estimation, and symbol detection.

In chapter 2, we introduce the basics of DL including supervised and unsupervised learnings, the architecture of different deep neural networks (DNNs), the backpropagation algorithm, and the autoencoders.

In chapter 3, we derive an upper bound on the expected generalization error of the convolutional neural network using information theoretical tools in terms of the entropy of the input and the number of layers. When there is no one-to-one mapping between hidden layers, the upper bound shows that as the number of hidden layers increases in the convolutional neural network, the expected generalization error decreases exponentially to zero. We showed that adding more layers is not rewarding when there is at least one one-to-one mapping between two consecutive hidden layers.

In chapter 4, we propose a learning-based approach relying on DNN to design a partial zero forcing (PZF) interference cancellation scheme in uplink cellular networks. Numerical results show that the learning-based approach mimics accurately the model-based approach while reducing substantially the execution time by more than 90 %.

In chapter 5, we develop a neural networks-based estimator that can explicitly estimate the channel state information (CSI) at the receiver without knowing the optical channel model. We design a neural networks-based detector that uses the estimated CSI to detect transmitted symbols without knowing the optical channel model.

1.5 Notation

Random variables are represented by uppercase letters, e.g., X, their realizations by lowercase letters, e.g., x, and their distributions by P_X. Random vectors are represented by bold capital letters, e.g., $\boldsymbol{X}$, their realizations by bold lower letters, e.g., $\boldsymbol{x}$, and their distributions by $P_{\boldsymbol{X}}$. We denote by $P_{(X,Y)}$ the distribution of the joint random variable (X, Y), and by $P_{Y|X}$ the distribution of the conditional random variable $Y|X$. $\underset{P_X}{\mathbb{E}}[.]$ denotes the expectation of the expression inside the brackets over random variable X.

CHAPTER 2

Learning-Based Communication Systems

2.1 Motivation

As we discussed in chapter 1, a typical wireless communication system contains a transmitter, channel, and receiver. The channel model characterizes the physical phenomena of the transmitted signals and helps in designing the transmitter and the receiver. Conventionally, traditional radio-frequency (RF) communication systems transmit signals at frequencies below 10 GHz. In these systems, the channel model follows tractable mathematical models that describe the propagation characteristics of the transmitted signals with reasonable accuracy. However, in other complex systems, it is difficult to find a tractable mathematical model that describes the channel model with high accuracy [2]. We highlight two complex scenarios where it is difficult to describe analytically the channel model. The first scenario is high frequency communications, i.e., millimeter-wave (mm-Wave) communications (30 GHz-100 GHz) [3]. The second scenario is underwater or molecular communications [4]. Mm-Wave channels offer massive expansion in bandwidth to support high data rates to users [5, 6, 7]. In mm-Wave communications, channel models are too complex and they change very fast. Hence, it is difficult to estimate the channels with reasonable accuracy. This estimation of CSI is necessary for many detection algorithms, i.e., ML, PZF, and LMMSE detection algorithms. The latter scenario includes molecular channels, i.e., the channel over which molecules transmit signal between each other to communicate [4]. In this category, channel models are unknown or difficult to derive. Therefore, it is difficult to use traditional detection algorithms to optimize the system performance.

DL is a subfield of machine learning that has recently showed unprecedented success in classification and prediction problems without the need of well-defined mathematical model [8]. It shines in many applications like computer vision, automatic speech recognition, and natural language processing [8]. Researchers are actively trying to extend this technology to other domains, such as wireless communications. Using DL techniques on a wide range of communication systems has achieved unprecedented success such as cognitive radio, resource management [9], link adaptation [10, 11] and positioning [12]. The DL-based communication system has promising applications in complex scenarios for the following reasons:

First, deep networks have shown an unprecedented ability to approximate diverse function [13], i.e., they are considered as universal function approximators, with high-level learning ability regardless of the complex channel conditions [2]. In DL-based communication systems, learned algorithms are represented by learned weights that optimize the end-to-end performance of the communication systems using convenient training methods instead of using well-defined and complex mathematical models.

Second, the distributed and parallel computing architectures of DL ensure computation speed and processing capacity. DL systems demonstrate a remarkable energy-efficiency through fast-developing parallelized processing architecture such as graphical processing units (GPUs).

2.2 Supervised and Unsupervised Learning

In this section, we present two learning concepts in DL including supervised and unsupervised learning.

2.2.1 Supervised Learning

In supervised learning, the learning algorithm learns the mapping function between the input space $\mathcal{X}$ and the output space $\mathcal{Y}$, i.e., $\mathcal{Y} = f(\mathcal{X})$. In this case, each training sample $Z_i = (X_i, Y_i)$, where $X_i \in \mathcal{X}$ is the input variable and $Y_i \in \mathcal{Y}$ is the label or the desired output variable. It is called supervised learning because the learning process of the underlying algorithm from the training dataset can be seen as a teacher supervising the learning process [14]. In other words, the algorithm knows the desired output of each input variable and tries to find the optimal mapping function that gives the closest solution to the label. The learning process stops when the learning algorithm achieves an acceptable level of performance. There are two types of supervised learning problems:

- **Classification:** A classification problem is when the output variable is a category. For instance, consider the problem of learning an algorithm for animal image classification. In this case, the output variable takes animal's names such as: "dog", "cat", and "lion".

- **Regression:** A regression problem is when the output variable is not discrete anymore but is real value, such as "weights" or "dollars".

Among popular examples of supervised learning algorithms, we have:

- DNN that is used to solve both classification and regression problems.

- Linear regression that is used to solve regression problems.

- Support vector machine that is used to solve classification problems.

2.2.2 Unsupervised Learning

In unsupervised learning, we have only the input variable $X \in \mathcal{X}$. In this case, each training sample $Z_i = X_i$ where $X_i \in \mathcal{X}$. The purpose of unsupervised learning is to let the learning algorithm learn the underlying distribution in the data in order to learn more about the data. It is called unsupervised learning because there are no correct answers, i.e., labels, and there is no teacher. Learning algorithms are left to their own to find the interesting distribution in the data [14]. A concrete example of unsupervised learning is a housekeeping robot. Nobody explicitly tells the robot what are the steps that it needs to follow to do the housework. The robot learns by itself to do the tasks. There are two types of unsupervised learning problems:

- **Clustering:** In clustering problems, the learning algorithm tries to find the inherent grouping in the data. For instance, we try to group customers depending on their purchasing behaviour.
- **Association:** In association problem, the learning algorithm tries to find rules that describe large portions of the data.

Among popular examples of unsupervised learning algorithms, we have:

- K-means that are used to solve clustering problems.
- Apriori algorithms that are used to solve association problems.

2.3 DNNs

In this section, we describe the architecture of two DNNs such as fully connected and convolutional neural networks.

2.3.1 An Overview of FNN

Below, we briefly explain how fully connected neural networks (FNNs) work. Essentially, it consists of 2 main engines. The first one is the construction of the underlying layers, with each layer consisting of a certain number of neurons (perceptron), weights and biases. The second consists of a backpropagation algorithm to update the weights and the biases.

The Structure of FNN

The FNN is an artificial neural network composed of many perceptrons and many layers [15]. For simplicity, we restrict to 3-layered FNN as showing in Fig. 2.1. The first, the second and the third layers are the input, the hidden and the output layer, respectively. In Figure 2.1, $\boldsymbol{x} = [x_1, \ldots, x_{n_I}]$ denotes the input of the FNN. Each input $\boldsymbol{x}$ corresponds to a label $\boldsymbol{y} = [y_1,\ y_2, \ldots,\ y_{n_o}]$. $a_i^{(k)}$ denotes the output for the i^{th} node in the k^{th} layer. $\boldsymbol{W^{(i)}}$

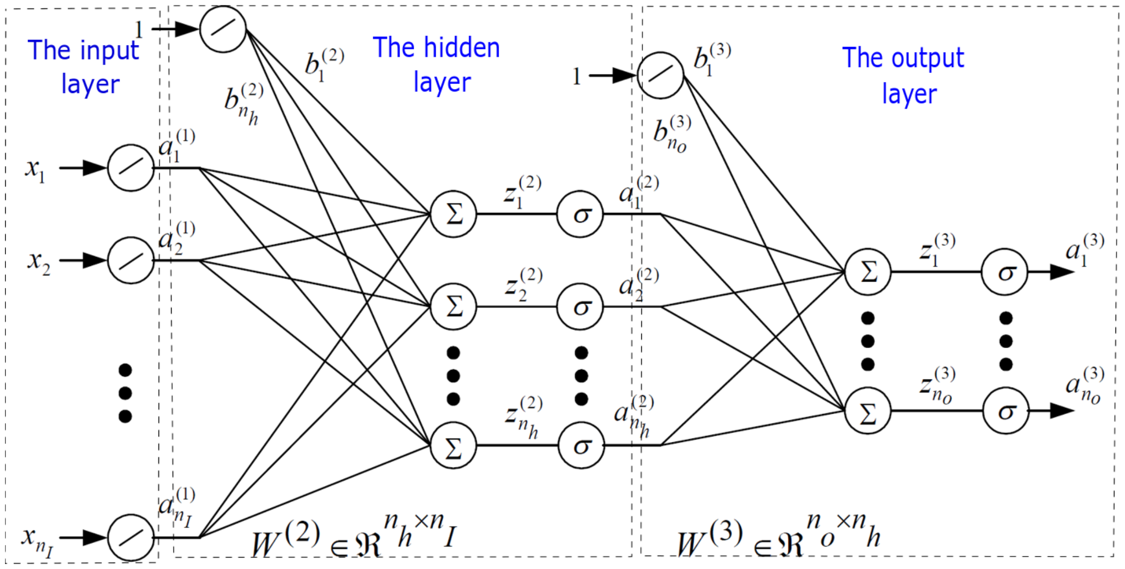


Figure 2.1: A 3-layered FNN. The input, the hidden and the output layers have n_I, n_H and n_o neurons, respectively.

contains the weight factors between the $(i-1)^{th}$ layer to the i^{th} layer. $b_i^{(k)}$ represents the bias for the i^{th} node in the k^{th} layer. σ is the activation function. $z_i^{(k)}$ denotes the output for the i^{th} node in the k^{th} layer before the activation function. For instance, for 3-layered FNN in Figure 2.1, the outputs of each layer are obtained as following

$$\boldsymbol{z}^{(i+1)} = \boldsymbol{W}^{(i+1)}\boldsymbol{a}^{(i)} + \boldsymbol{b}^{(i+1)}, \tag{2.1}$$

$$\boldsymbol{a}^{(i+1)} = \sigma\left(\boldsymbol{z}^{(i+1)}\right), \ i = 1, \ 2, \tag{2.2}$$

where $\boldsymbol{a}^{(i)} = [a_1^{(i)}, \ldots, a_n^{(i)}]^T$, $\boldsymbol{b}^{(i)} = [b_1^{(i)}, \ldots, b_n^{(i)}]^T$ and $\boldsymbol{z}^{(i)} = [z_1^{(i)}, \ldots, a_n^{(i)}]^T$, with $n \in \{n_I, n_h, n_o\}$. The objective of the FNN is to obtain a final output ($\boldsymbol{a}^{\mathbf{(3)}}$ in this case) as close as possible to the label $\boldsymbol{y}$ corresponding to the current processed input sample $\boldsymbol{x}$.

The Backpropagation Algorithm

The main function of the backpropagation algorithm is to update the weights and the biases to optimize the following quadratic cost function [15] (Applied in the FNN in

Figure 2.1)

$$C = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{a}^{(3)}\|^2 = \frac{1}{2}\sum_{j=1}^{n_o}\left(y_j - a_j^{(3)}\right)^2. \tag{2.3}$$

The backpropagation algorithm is a systematic way to compute

$$\frac{\partial C}{\partial w_{ij}^{(3)}},\ \frac{\partial C}{\partial b_j^{(3)}} \text{ for } i = 1, 2, \ldots, m,\ j = 1, 2, \ldots,\ n, \tag{2.4}$$

where $n = n_h$ if $m = n_o$ and $n = n_I$ if $m = n_h$; $w_{ij}^{(k)}$ denotes the weight between the j^{th} node, in the $(k-1)^{th}$ layer, and the i^{th} node in the $(k)^{th}$ layer. For quadratic cost function, the backpropagation algorithm is presented as follows:

$$\begin{cases} \frac{\partial C}{\partial \boldsymbol{W}^{(3)}} = \boldsymbol{\delta}^{(3)}\boldsymbol{a}^{(2)T}, \\ \frac{\partial C}{\partial \boldsymbol{b}^{(3)}} = \boldsymbol{\delta}^{(3)}, \\ \boldsymbol{\delta}^{(2)} = \boldsymbol{W}^{(3)T} \odot \sigma'\left(\boldsymbol{z}^{(2)}\right), \\ \frac{\partial C}{\partial \boldsymbol{W}^{(2)}} = \boldsymbol{\delta}^{(2)}\boldsymbol{a}^{(1)T}, \\ \frac{\partial C}{\partial \boldsymbol{b}^{(2)}} = \boldsymbol{\delta}^{(2)}, \end{cases} \tag{2.5}$$

where $\boldsymbol{\delta}^{\mathbf{(3)}}$ is given by

$$\boldsymbol{\delta}^{\mathbf{(3)}} = \left[\left(a_1^{(3)} - y_1\right)\sigma'\left(z_1^{(3)}\right), \cdots, \left(a_{n_o}^{(3)} - y_{n_o}\right)\sigma'\left(z_{n_o}^{(3)}\right)\right], \tag{2.6}$$

where $\sigma'(\cdot)$ is the derivative of the activation function $\sigma(\cdot)$ and $\odot$ is the Hadamard product. Backpropagation algorithm is based around 4 fundamentals equations (2.5), i.e., the partial derivatives of weights and biases. In each iteration, the FNN implements the backpropagation algorithm in order to optimize the cost function (2.3), i.e., to converge toward the desired level $\boldsymbol{y}$.

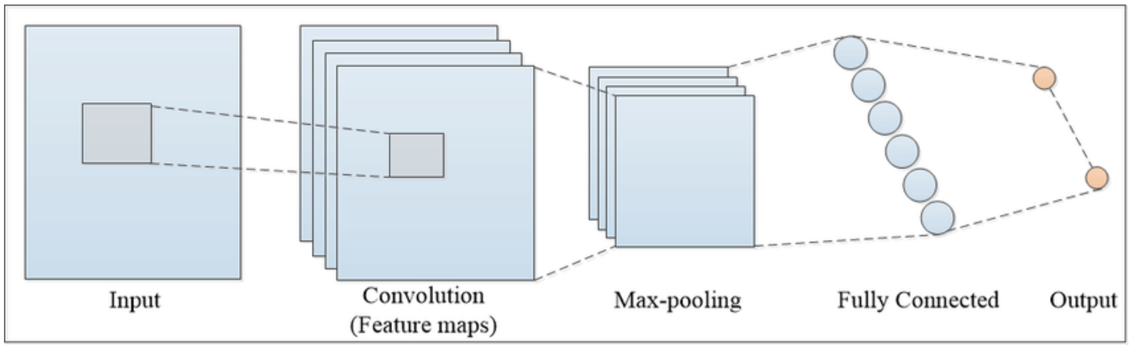


Figure 2.2: A 5-layered CNN: The input, the convolution, the pooling, the fully connected, and the output layers.

2.3.2 An Overview of CNN

In this section, we describe the layers of convolutional neural network (CNN). As can be seen in Fig. 2.2, CNNs have three types of layers: convolutional, pooling and fully connected layers. A convolutional layer computes the output of a convolution operation between the input image and a set of learnable filters. The benefit of using this layer is that the convolution operation is not over the whole input image, but over a specific region called receptive field. To process all the input image's pixels, the filter shifts with a known stride parameter. Hence, we obtain a map feature with less dimensionality compared to the input image. Pooling layers are also referred to as downsampling layers. These layers come after the convolutional layer. They reduce the spatial dimension of the feature map by taking a filter of size 2×2 and a stride of length 2. There are two types of pooling layers: max pooling and average pooling. Max pooling is applied to the feature map and it outputs the maximum number in every sub-region that the filter convolves around. Average pooling outputs the average number in every sub-region that the filter convolves around. Pooling layers serve two purposes. The first is that the number of weight parameters is reduced, and this lessens the complexity cost. The second is that it avoids the overfitting [16] which happens when the CNN performs well on the training data, but it performs poorly on the testing data. Fully connected layers are located at the end of CNN.

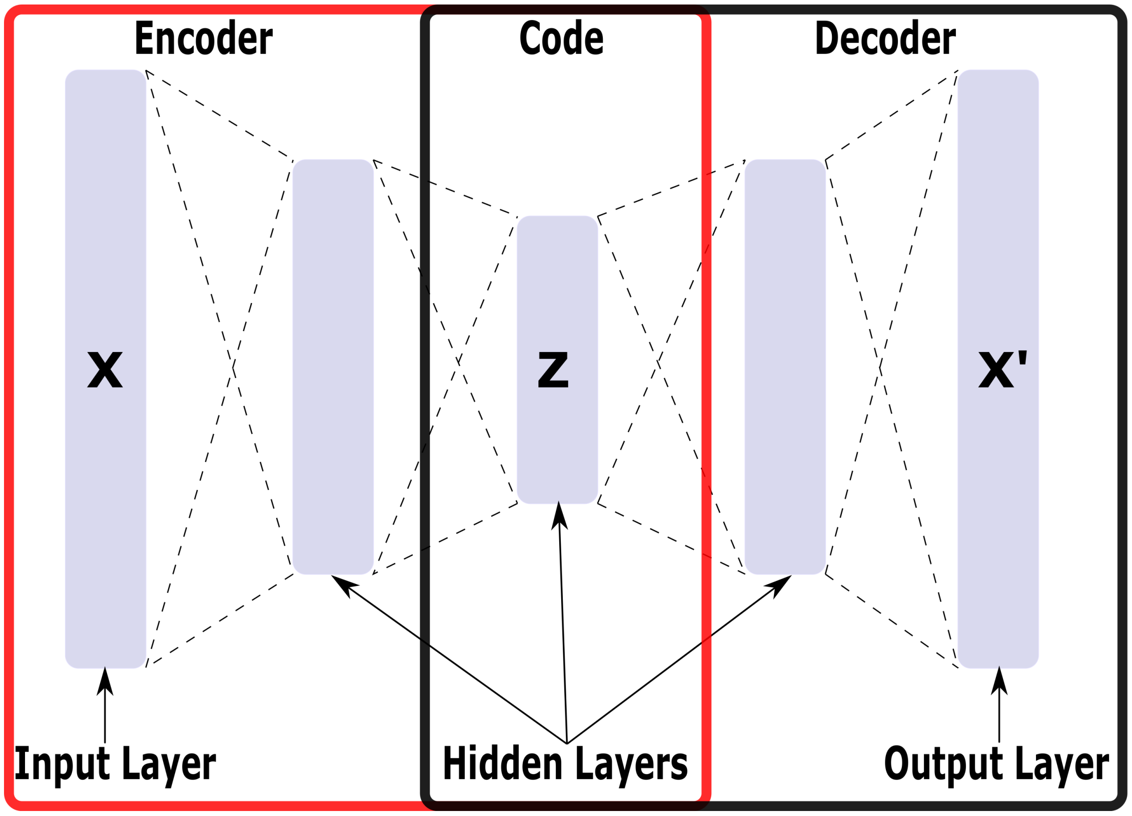


Figure 2.3: The structure of an AE with three fully connected hidden layers.

2.4 Autoencoders

Autoencoders (AEs) are neural networks (NNs) that are used to solve unsupervised learning problems [14]. AEs are trained to attempt to copy its input to its output [14]. They are used for data's dimensionality reduction [17]. The structure of an AE is shown in Fig. 2.3. As can be seen in Fig. 2.3, an AE consists of two parts, the encoder and the decoder. The encoder and decoder are defined as transition functions $\phi : \mathcal{X} \rightarrow \mathcal{F}$ and $\psi : \mathcal{F} \rightarrow \mathcal{X}$, respectively, where $\mathcal{X}$ is the input space and $\mathcal{F}$ is the feature space. The AE attempts to copy its input to its output by solving the following optimization problem:

$$\phi, \psi = \underset{\phi,\psi}{\operatorname{argmax}} \left\| \boldsymbol{x} - (\psi \circ \phi)\, \boldsymbol{x} \right\|^2, \tag{2.7}$$

where $\boldsymbol{x} \in \mathcal{X}$ is the input vector to the AE. In the simplest case, when the AE contains three layers: an input, a hidden, and an output layers, respectively, the encoder part of the AE takes the input $\boldsymbol{x} \in \mathbf{R}^d = \mathcal{X}$ and maps to $\boldsymbol{z} \in \mathbf{R}^p = \mathcal{F}$, where d and p denotes the number of neurons in the input and the hidden layers, respectively. The mapping between $\boldsymbol{x}$ and $\boldsymbol{z}$ is given by:

$$\boldsymbol{z} = \sigma\left(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}\right), \tag{2.8}$$

where $\boldsymbol{W}$ is the matrix weight, $\boldsymbol{b}$ is the bias vector, and $\sigma\left(\cdot\right)$ is an element-wise activation function. The image $\boldsymbol{z}$ is called the code, latent variables, or latent representation [14]. The decoder part of the AE takes the code $\boldsymbol{z}$ and maps to the reconstruction vector $\boldsymbol{x}' \in \mathbf{R}^d = \mathcal{X}$ which has the same dimension as the input vector $\boldsymbol{x}$. This mapping is given by

$$\boldsymbol{x}' = \sigma'\left(\boldsymbol{W}'\boldsymbol{z} + \boldsymbol{b}'\right), \tag{2.9}$$

where $\boldsymbol{W}'$ is the matrix weight, $\boldsymbol{b}'$ is the bias vector, and $\sigma'\left(\cdot\right)$ is an element-wise activation function of the decoder. $\boldsymbol{W}'$, $\boldsymbol{b}'$, and $\sigma'\left(\cdot\right)$ may differ from $\boldsymbol{W}$, $\boldsymbol{b}$, and $\sigma\left(\cdot\right)$ of the encoder which depends on the design of the AE. During the training stage, the AE attempts to minimize the reconstruction error/loss, which is given as follows:

$$\mathcal{L}\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \|\boldsymbol{x} - \boldsymbol{x}'\|^2 = \|\boldsymbol{x} - \sigma'\left(\boldsymbol{W}'\left(\sigma\left(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}\right)\right) + \boldsymbol{b}'\right)\|. \tag{2.10}$$

CHAPTER 3

Upper Bound on the Expected Generalization Error of the Convolutional Neural Network

3.1 Motivation and Related Work

DL is a subfield of machine learning (ML) that has capabilities in classifications and predictions [18]. It shines in many applications like computer vision and natural language processing because it is really difficult to model real images and languages using an accurate mathematical function. CNNs have been used in image recognition [16] and they showed a great success due to the large public image repositories, such as Imagenet and Quickdraw [19, 20], and high-performance computing systems, such as GPUs. However, despite their success, there is still no in-depth theoretical understanding of their learning and optimization process. Understanding the inner mechanism in DL has been an open problem for many years and much effort is being deployed to explain its success. Among the open questions that researchers are trying to answer are: Why NNs are able to predict outcome values for previously unseen data? To answer that question, many works tried to bound the generalization error. It is the difference between the expected loss of the testing data and the empirical loss of the training data [21, 22]. This measure is used to quantify the degree to which a supervised DL algorithm may overfit to training data, i.e., how the learning algorithm fits well on testing data [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]. Bounding the generalization error provides a theoretical understanding of the DL algorithms' performance so that, ultimately, it helps designing better learning algorithms.

Research works tried to understand the learning process of learning algorithms by bounding the generalization error which reflects how the learning algorithm fits well on testing data, i.e, an unseen data. For example, the works in [21, Chap. 8],[22, Chap. 6] obtained an upper bound over the Vapnik-Chervonenkis (VC) dimension. The authors in [21, 22] used the VC dimension to ensure that the empirical risk will converge uniformly to the expected risk.

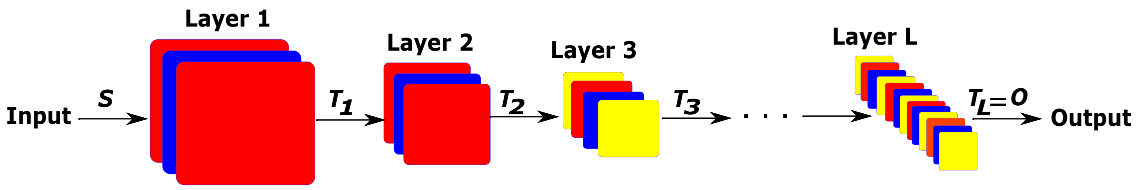


Figure 3.1: A CNN with L hidden layers. $\boldsymbol{S}$ represents the input of the CNN, $\boldsymbol{T}_k$ represents the output of the k^{th} hidden layer, $k \in \{1, \cdots, L\}$. The output of the L^{th} hidden layer is equal to the output of the CNN, i.e., $\boldsymbol{T}_L = \boldsymbol{O}$.

The authors in [23] obtained bounds on the generalization error of support vector machines (SVMs) both in the classification and in the regression framework that do not depend on the implicit VC dimension. In [24], an energy-based exploration of random features (EERF) algorithm is proposed to maintain a low generalization error in supervised learning. The authors in [24] obtained an upper bound on the generalization error of the EERF algorithm. Moreover, [25] introduces new stability-based and information-theoretical tools for analysis of the generalization of learning algorithms in the adaptive setting. The authors in [26, 27] proposed a margin based generalization bound. The bound in [27] depends on the elementwise $L1$-norm of the weights in each layer, while the bound derived in [26] depends on the Frobenius norm (elementwise $L2$-norm) of the weights in each layer. Recent work [28] discusses generalization error bounds of learning algorithms in the probably approximately correct (PAC) model based on mutual information. The authors in [25, 28] assumed that the output has low mutual information with the input dataset to derive those bounds. In [29], the authors derived an upper bound on the generalization error of a learning algorithm in terms of the mutual information between the input and the output. The authors in [29] restrict the loss function to be a σ-sub-Gaussian to derive their upper bound. In [30], the authors propose several information-theoretic measures and use them to upper-bound the generalization error of learning algorithms. The framework in [30] is complementary to the information-theoretic methodology developed in [29]. Finally, the authors in [31] obtained an upper bound of the generalization error in terms of the mutual information when the loss function has an upper bounded cumulant generating function.

3.2 An Information-Theoretical View of CNN

In this section, we present the relationship between CNN and the concept of information loss. We formally define the generalization error.

3.2.1 CNN and Information Loss

Figure 3.1 infers that a CNN can be seen as a Markov chain. In other words, $\boldsymbol{S} \rightarrow \boldsymbol{T}_1 \rightarrow \cdots \boldsymbol{T}_{L-1} \rightarrow \boldsymbol{T}_L = \boldsymbol{O}$ forms a Markov chain. Using data processing inequality (DPI)[32], we have the following relationships:

$$\begin{aligned} & I\left(\boldsymbol{S};\boldsymbol{O}\right) = I\left(\boldsymbol{S};\boldsymbol{T}_L\right) \\ \leq\ & I\left(\boldsymbol{S};\boldsymbol{T}_{L-1}\right) \leq I\left(\boldsymbol{S};\boldsymbol{T}_{L-2}\right) \leq \cdots \leq I\left(\boldsymbol{S};\boldsymbol{T}_1\right) \\ \leq\ & I\left(\boldsymbol{S};\boldsymbol{S}\right) = H\left(\boldsymbol{S}\right), \end{aligned} \tag{3.1}$$

where $H\left(\boldsymbol{X}\right)$ represents the entropy of the random variable $\boldsymbol{X}$ and $I\left(\boldsymbol{X};\boldsymbol{Y}\right)$ denotes the mutual information between random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$. Equation (3.1) clearly states that the output of each layer can only get nosier as we progress towards the CNN's output $\boldsymbol{O}$. Equivalently, the information about $\boldsymbol{S}$ gained by observing the output's layer can only decrease while moving from one layer to another towards the CNN's output.

3.2.2 Strong Data Processing Inequality

In this subsection, we introduce the concept of strong data processing inequality (SDPI)[33] which is given in the following Theorem:

Theorem 1. *[33] When three random variables X, Y and Z form a Markov chain, i.e., $X \rightarrow Y \rightarrow Z$, and if the mapping $P_{Z|Y}$ is noisy, i.e., we can not recover Y perfectly from* *the observed random variable Z, then there exists $0 \leq \eta < 1$, such that:*

$$I\left(X;Z\right) \leq \eta I\left(X;Y\right). \tag{3.2}$$

From Fig. 3.1, we have $\boldsymbol{T}_{j-2} \rightarrow \boldsymbol{T}_{j-1} \rightarrow \boldsymbol{T}_j$ form a Markov Chain, $\forall j \in \{3, \cdots, L\}$. Since the mapping $P_{\boldsymbol{T_j}|\boldsymbol{T_{j-1}}}$ is not necessarily noisy, then (3.2) could be replaced by:

$$I\left(\boldsymbol{T}_{j-2};\boldsymbol{T}_j\right) \leq \eta I\left(\boldsymbol{T}_{j-2};\boldsymbol{T}_{j-1}\right), \tag{3.3}$$

where $0 \leq \eta \leq 1$. Note that η in (3.3) is equal to 1 if and only if $\boldsymbol{T}_j$ and $\boldsymbol{T}_{j-1}$ are linked via a one-to-one mapping. Applying this propriety recursively to the Markov chain in Fig. 3.1, we obtain the following relationships:

$$\begin{aligned} & I\left(\boldsymbol{S};\boldsymbol{O}\right) = I\left(\boldsymbol{S};\boldsymbol{T}_L\right) \\ \leq \;& \eta_L I\left(\boldsymbol{S};\boldsymbol{T}_{L-1}\right) \leq \eta_L \eta_{L-1} I\left(\boldsymbol{S};\boldsymbol{T}_{L-2}\right) \\ \leq \;& \cdots \leq \left(\prod_{k=1}^{L} \eta_k\right) I\left(\boldsymbol{S};\boldsymbol{S}\right) = \left(\prod_{k=1}^{L} \eta_k\right) H\left(\boldsymbol{S}\right), \end{aligned} \tag{3.4}$$

where $0 \leq \eta_k \leq 1$, $k = 1, \ldots, L$, quantifies the information loss in the k^{th} hidden layer. Note again that $\prod_{k=1}^{L} \eta_k = 1$ if and only if $\eta_k = 1$, $\forall k \in \{1, \cdots, L\}$. This means that the mapping between hidden layers is a one-to-one mapping.

3.2.3 The Generalization Error

A CNN $\mathcal{F} : \mathcal{Z} \rightarrow \mathcal{W}$ can be regarded as a randomized mapping from the training sample space $\mathcal{Z}$ to the hypothesis space $\mathcal{W}$. In other words, if we denote this randomized mapping by $P_{W|\boldsymbol{S}}$, then the CNN takes as input a training sample $\boldsymbol{S} = [Z_1, Z_2, \cdots, Z_n]$ of size n of independent and identically distributed (i.i.d.) random elements of $\mathcal{Z}$ drawn from unknown distribution P_Z, and picks a random element W of the hypothesis space $\mathcal{W}$. We denote by $l : \mathcal{Z} \times \mathcal{W} \rightarrow [0, 1]$ the loss function to measure the quality of a prediction with respect to a hypothesis. We assume that the loss function is bounded. The true risk of a hypothesis $W \in \mathcal{W}$ on P_Z is defined as follows [30]:

$$L_{P_Z}(W) \triangleq \underset{P_Z}{\mathbb{E}}\left[l(W, Z)\right]. \tag{3.5}$$

Since P_Z is unknown, the learning algorithm cannot directly compute $L_{P_Z}(W)$ for any $W \in \mathcal{W}$, but can instead compute the empirical risk of W as a proxy which is defined as follows [30]:

$$L_{\boldsymbol{S}}(W) \triangleq \frac{1}{n}\sum_{i=1}^{n} l(W, Z_i). \tag{3.6}$$

In practice, the true risk is just approximated with a test data [34]. In other words, the empirical test risk is used to approximate the true risk [34]. For a CNN characterized by $P_{W|\boldsymbol{S}}$, the generalization error on P_Z is the difference $L_{P_Z}(W) - L_{\boldsymbol{S}}(W)$, and its expected value is given as follows [30]:

$$\begin{aligned} & GEN\left(P_Z, P_{W|\boldsymbol{S}}\right) \\ = \; & \underset{P_{(\boldsymbol{S},W)}}{\mathbb{E}}\left[L_{P_Z}(W) - L_{\boldsymbol{S}}(W)\right] && (3.7) \\ = \; & \underset{P_{(\boldsymbol{S},W)}}{\mathbb{E}}\left[\underset{P_Z}{\mathbb{E}}\left[l(W, Z)\right] - \frac{1}{n}\sum_{i=1}^{n} l(W, Z_i)\right], && (3.8) \end{aligned}$$

where the expectation is taken with respect to the joint distribution $P_{(\boldsymbol{S},W)}$. A small expected generalization error implies that the learned hypothesis will have similar performance on both the training and testing datasets, i.e., there is no overfitting.

3.3 Main Result

In this section, we present the main result which is an upper bound on the expected generalization error of CNNs as formalized in the following theorem:

Theorem 2. *The expected generalization error for a CNN with L hidden layers, input $\boldsymbol{S}$*

with distribution $P_{\boldsymbol{S}}$, and output W, can be upper bounded as follows:

$$|GEN\left(P_Z, P_{W|\boldsymbol{S}}\right)| \leq \exp\left(-\frac{L}{2}\log\frac{1}{M}\right)\sqrt{\frac{2\log(2)\, H(\boldsymbol{S})}{n}}, \tag{3.9}$$

where $0 \leq M = \max_{k=1,\cdots,L} \eta_k \leq 1$, with η_k quantifies the information loss in the k^{th} hidden layer.

We note that the results in Theorem 2 provide an analytical closed-form upper bound on the expected generalization error of the CNN. Below, we provide insights as to how Theorem 2 could be used to understand the learning process of the CNN. For example, when there is at least one one-to-one mapping between two consecutive hidden layers, then $M = 1$, and the upper bound does not depend on the number of hidden layers L. Therefore, introducing more layers is not rewarding in this case. When there is no one-to-one mapping between hidden layers, then $M < 1$ and the expected generalization error decreases exponentially to zero as the number of hidden layers L increases in the CNN, because $\log\left(\frac{1}{M}\right) > 0$. Hence, adding more layers is beneficial only if all the mapping between any two consecutive layers is noisy [1], from a generalization error point of view.

3.4 Numerical Results

In this section, we study the performance of the proposed CNN on the classification of Quickdraw dataset. First, we describe the dataset. Second, we present the CNN architecture that we used in our simulation. Then, we describe how we implemented the CNN. Finally, we present the obtained results.

[1] In practice, each hidden layer of CNN can be a noisy version of the previous hidden layer when the mapping between these two hidden layers is not a one-to-one mapping, i.e., not an invertible mapping. For instance, the rectified linear unit (ReLU) activation function is not a one-to-one mapping function. All the negative values in the ReLU have the same output which is a zero output.

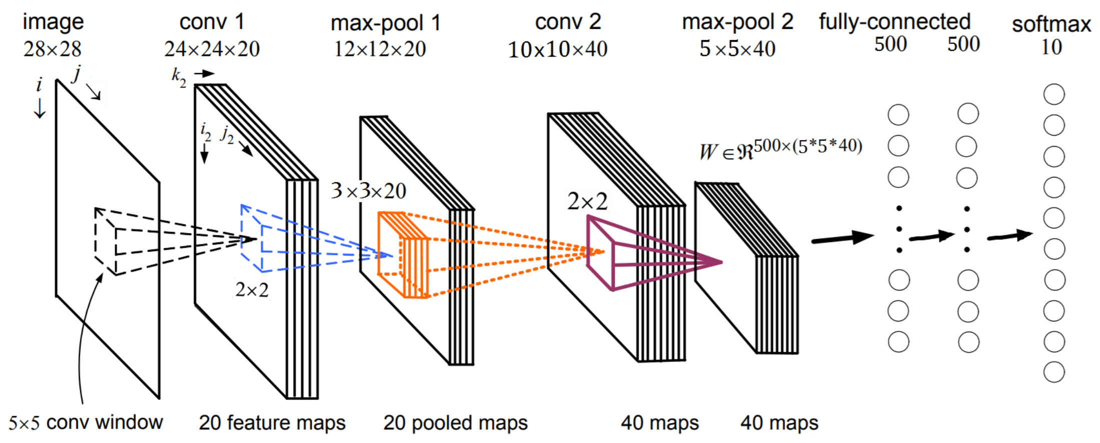


Figure 3.2: The architecture of the CNN.

3.4.1 Quickdraw Dataset

we use a CNN to classify Quickdraw images, i.e., a human-drawn images containing different classes of objects released by Google. Quickdraw consists of 345 classes. From 345 categories, we restricted to classify only 10 categories of images which are: Alarm clock, Aircraft-carrier, Apple, Calendar, Cake, Crab, Hurricane, Purse, Sink, and Banana.

3.4.2 CNN Architecture

As can be seen in Fig. 3.2, the architecture of the CNN contains $L = 7$ layers: two convolutional layers, two max-pool layers, two fully connected layers, and an output layer. The first convolutional layer contains 20 feature maps with size 24×24, while the second convolutional layer contains 40 filters with size 10×10. Every fully connected layer contains 500 neurons. The number of outputs for the output layer is 10, as we have 10 categories for classification.

3.4.3 The Implementation

We train our CNN over 50,000 samples of training set in batch sizes of 10. We set the number of iterations, i.e., the number of epochs, to 40. We use a mean square error (MSE) as a lost function and stochastic gradient descent (SGD) to train our CNN. After the training, we test it over 1,000,000 samples. We implemented the training and testing phase using Theano [35].

3.4.4 Results

As shown in Fig. 3.3, The network had an accuracy of 91.2% on the testing data. It was the best accuracy reached during several trials. During the learning process the network started with 74% and then learned fast and reached 80% within the first epoch. By epoch 15 the network reached the 90% and in epoch 35 it reached the best test accuracy.

Figure 3.4 illustrates the expected generalization error for different number of hidden layers L. It shows the expected generalization error and the upper bound. To plot the expected generalization error which is represented by the red curve, we computed the difference between the empirical testing error and the empirical training error for different samples set of size n. After that, we computed the average. To plot the upper bound given by (3.6) that is represented by the black curve in Fig. 3.4, we have to first compute $H(\boldsymbol{S})$ as well as M. To compute $H(\boldsymbol{S})$, we use the method used by [36]. In this paper, they used an entropy estimator developed by Kraskov et al. [37] and they uploaded the correspondent code in Github [38]. The built-in package that calculates the entropy is called non-parametric entropy estimation toolbox (NPEET) [39]. However, to compute M, we first evaluate the ratios of mutual information of each pair of consecutive hidden layer. We then find the maximum value of these ratios numerically and plug it as M in (3.6). To compute the mutual information between any two successive hidden layers, we use the same method used by [36]. In this paper, they used a mutual information estimator developed by Kraskov et al. [37] and

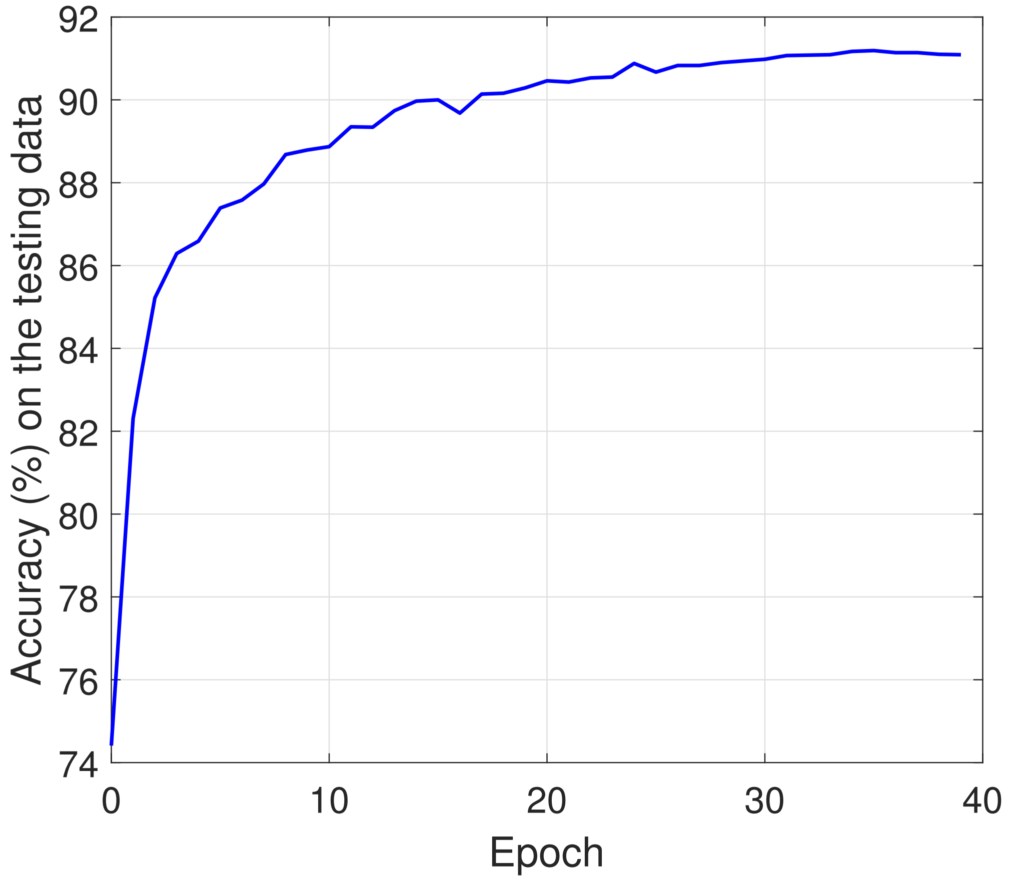


Figure 3.3: The learning progress of the CNN.

they uploaded the correspondent code in Github [40]. The same built-in package NPEET calculates the mutual information [39]

Remark 1: If for some layers the information loss is equal to 1, i.e., $\eta_k = 1$ for some $1 \leq k \leq L$, then the k^{th} layer does not provide any information gain. Hence, we can remove that layer from the CNN.

Figure 3.4 shows that when the number of hidden layers L increases, the gap between the upper bound and the expected generalization error decreases. As shown in Fig. 3.4, the expected generalization error decreases rapidly when the number of hidden layers L increases. For instance, training the CNN with $L = 15$ hidden layers drops the expected generalization error by 80% compared to training the CNN with $L = 7$ hidden layers. Therefore, increasing the number of hidden layers reduces the overfitting which is confirmed analytically by the

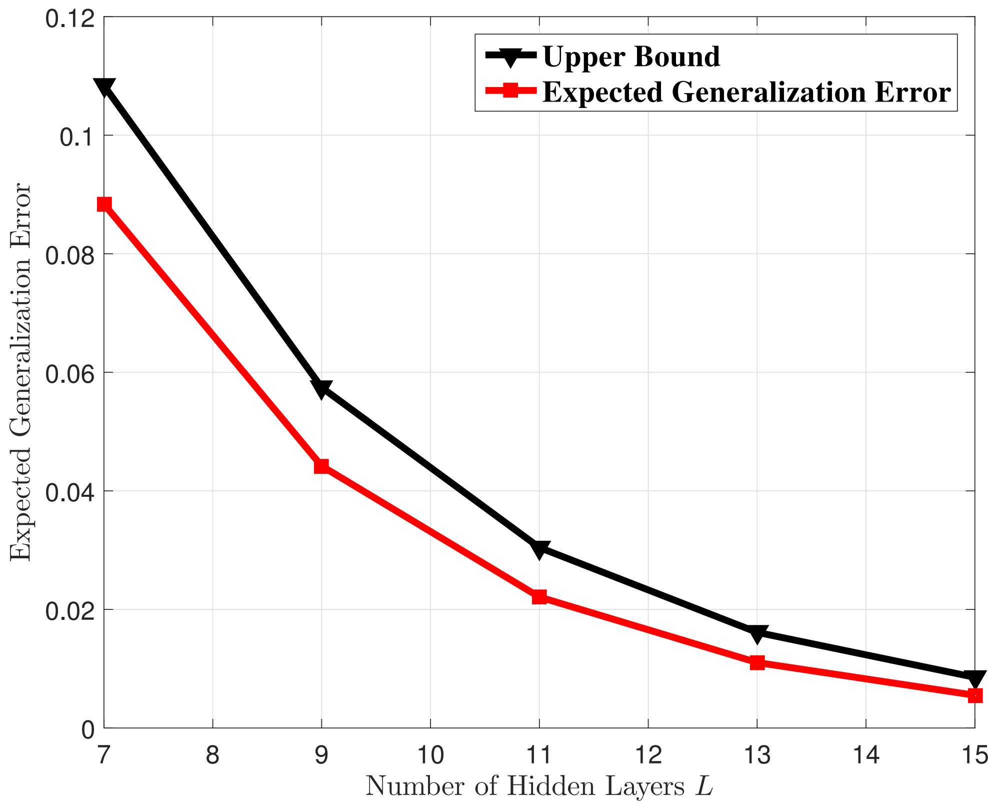


Figure 3.4: The expected generalization error versus the number of hidden layers L. The black and red curves represent the upper bound and the expected generalization error, respectively. To vary the number of hidden layers L, we added a fully connected layer in each case which contains 500 neurons.

tightness of the upper bound on the generalization error for L large enough.

CHAPTER 4

Training Deep Neural Networks for Partial Interference Cancellation in Uplink Cellular Networks

4.1 Motivation and Related Work

In wireless communications systems, partial interference cancellation (PIC) has been conventionally performed using analytical methods assuming that an accurate channel model is available. However, in practical communications systems, such accurate models may not be always easy to devise. In this case, it is difficult to derive performance limits for realistic wireless channels. Furthermore, even when an accurate channel model exists, PIC induces a high implementation complexity which compromises its benefit.

Motivated by the success of NNs in various fields, we propose a learning-based PZF interference cancellation scheme to overcome the complexity issue and demonstrate the efficiency of the proposed approach numerically. Specifically, we use DNN [41] to mimic the PZF algorithm [42]. The choice of DNN to perform this task is motivated by the fact that it only requires simple matrix-vector multiplications which reduces the complexity enormously. Another reason of using DNNs is their energy-efficiency when implemented on GPUs, thus providing real time processing [43, 44].

Recently, they have been several works suggesting that learning-based algorithms can accomplish as good as model-based counterparts in communications systems. In [42], the authors show that DNNs can estimate the CSI and can detect the symbols in an orthogonal-frequency division multiplexing (OFDM) system. In [45], a recurrent neural network (RNN) is proposed to decode, over an AWGN channel, a sequence of bits that has been encoded using linear codes. The authors in [2] propose an AE-based communications system, in which, the transmitter and the receiver are FNNs with multiple dense layers. The authors proved through extensive simulations that the proposed method can achieve the optimal accuracy of a Hamming code with ML decoding. The authors in [46] showed that it is possible to design an AE-based point to point communications system that can be used for over-the-air transmissions. In [47], an AE is developed to transmit and detect reliably over a Rayleigh fading channel without having to equalize for the channel gains.

4.2 System Model

The network comprises $K + 2$ nodes. It includes a base station (BS), a reference transmitter and K interferers. We chose a finite number of interferers ($K < \infty$) to capture cellular networks with finite number of interferers. Each transmitter has one single antenna while the BS has M receive antennas. The receiver has perfect CSI of N_c interferers and only knows the statistics of the remaining $K - N_c$ interferers. The BS, then, decodes the first set via a PIC approach while treating the second set of interferers as noise [48]. The signal at each receive antenna is corrupted by an AWGN with zero mean and variance σ^2. During the transmission process and at an instant t, the i^{th} node transmits the signal $s_i\left(t\right)$, with $E\left[s_i\left(t\right)^2\right] = P_i$, the transmitted power of user i. The channel vector between the i^{th} transmitter and the BS is equal to $\boldsymbol{h}_i = [h_{1i}, h_{2i}, ..., h_{Mi}]^T$ where h_{ji} is the complex channel coefficient between the i^{th} transmitter and the j^{th} receive antenna. Each channel gain $\boldsymbol{h}_i$ accounts for Rayleigh fading and shadowing as described by $\boldsymbol{h}_i = 10^{\frac{\eta_i}{20}}\boldsymbol{w}_i,\ 0 \leq i \leq K$, where η_i is the shadowing factor and $\boldsymbol{w_i} \sim \mathcal{CN}\left(\boldsymbol{0}, \boldsymbol{I}_M\right)$, is the fading vector. In the presence of log-normal shadowing, the $\{\eta_i\}$ are i.i.d. Gaussian with mean μ_s and variance σ_s^2. We denote by $\boldsymbol{x}(t) = [x_1(t), x_2(t), ..., x_M(t)]^T$ the vector of signals at M receive antennas.

$$\boldsymbol{x}(t) = \sum_{i=0}^{N_c} \boldsymbol{h}_i s_i(t) + \left(\sum_{i=N_c+1}^{K} \boldsymbol{h}_i s_i(t) + \boldsymbol{n}(t) \right). \tag{4.1}$$

4.3 Model-Based PZF Scheme

Now, the receiver may utilize a PZF decoding strategy to cancel the N_c interferers [49, 48]. Accordingly, let us define $\boldsymbol{H}\left(N_c\right) = \left(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{N_c}\right) \in \mathbb{C}^{M \times N_c}$, which includes the effective channels from the N_c interfering nodes. Departing from (4.1), PZF receiver cancels N_c interferers by multiplying the received signal $\boldsymbol{x}(t)$ by a filter $\boldsymbol{v}(N_c)^H$ given by:

$$\boldsymbol{v}(N_c) = \frac{\left(\boldsymbol{I} - \boldsymbol{H}(N_c)\,\boldsymbol{H}(N_c)^{\dagger}\right)\boldsymbol{h}_0}{\left\|\left(\boldsymbol{I} - \boldsymbol{H}(N_c)\,\boldsymbol{H}(N_c)^{\dagger}\right)\boldsymbol{h}_0\right\|}, \tag{4.2}$$

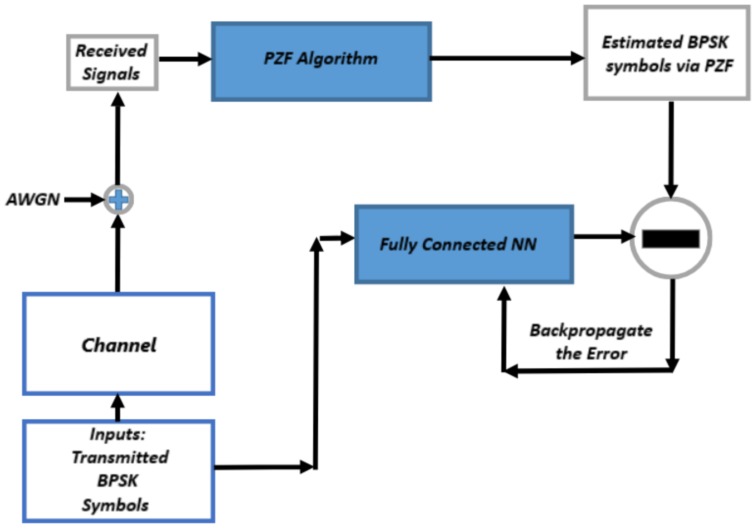
where $\boldsymbol{A}^{\dagger}$ designates the pseudoinverse of $\boldsymbol{A}$. Observe that $\boldsymbol{v}(N_c)$ in (4.2) reduces to maximum ratio combining (MRC) when $N_c = 0$ and to full ZF when $N_c = M - 1$.

4.4 Learning-Based PZF Scheme

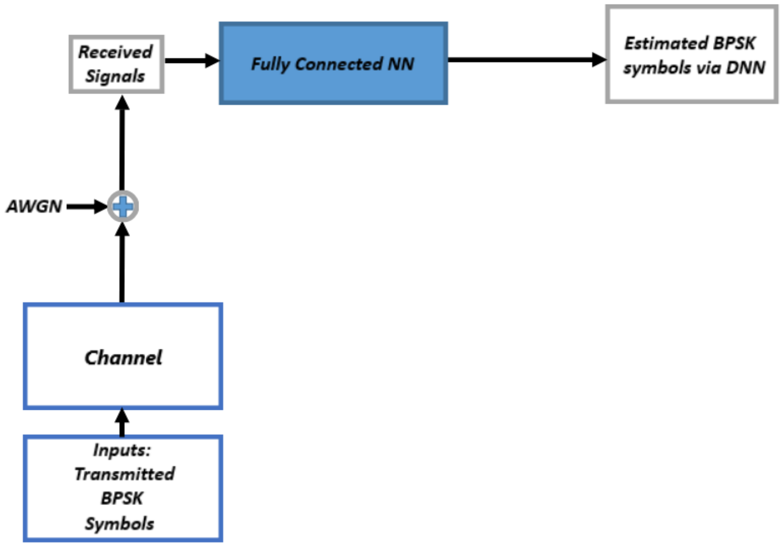
In this section, we present the proposed approach that uses DNN to mimic PZF. First, we present a description of the DNN architecture. Second, we show how we generate the training and the testing data. Finally, we explain how training and testing stages are performed.

4.4.1 The DNN architecture

As can be seen in Fig. 4.2, our FNN contains one input layer, three hidden layers, and one output layer. The received signals $\boldsymbol{x} \in \mathbb{C}^{M\times 1}$ are the inputs of the network. Hence, the input layer contains M neurons. The outputs of the network are the estimates of the desired transmitter's binary phase shift keying (BPSK) symbols via DNN $\hat{s}_{0,DNN}$. Therefore, the output layer contains one neuron. The first, the second and the third hidden layers contain 200, 80, and 80 neurons, respectively. A ReLU function is used as the activation function for all hidden layers, and a tangent hyperbolic (Tanh) activation function is used as the activation function of the output layer. The reason why we chose the fully connected (FC) architecture instead of the CNN architecture is that the former is more performing in terms of prediction and classification accuracy than the latter when the data inputs are independent from each other [14]. Note that the CNN architecture is more suitable when the data inputs are correlated [14]. For instance, CNNs are more accurate in image classification because



(a) Training stage.


Received Signals
Fully Connected NN
Estimated BPSK symbols via DNN
AWGN
Channel
Inputs: Transmitted BPSK Symbols

(b) Testing stage.

Figure 4.1: DNN-based PZF scheme. (a) In the training stage, a fully-connected neural network NN is trained using samples obtained from model-based PZF developed in section III. (b) In the testing stage, the trained network is incorporated into the receiver to mimic PZF and estimate the BPSK symbols.

the image itself contains finite number of pixels that are totally correlated to each other. In our case, we assume the received signals at the BS are totally independent from each other over time, i.e, there is no time correlation in the received signal. Hence, FC architecture is more efficient than CNN architecture.

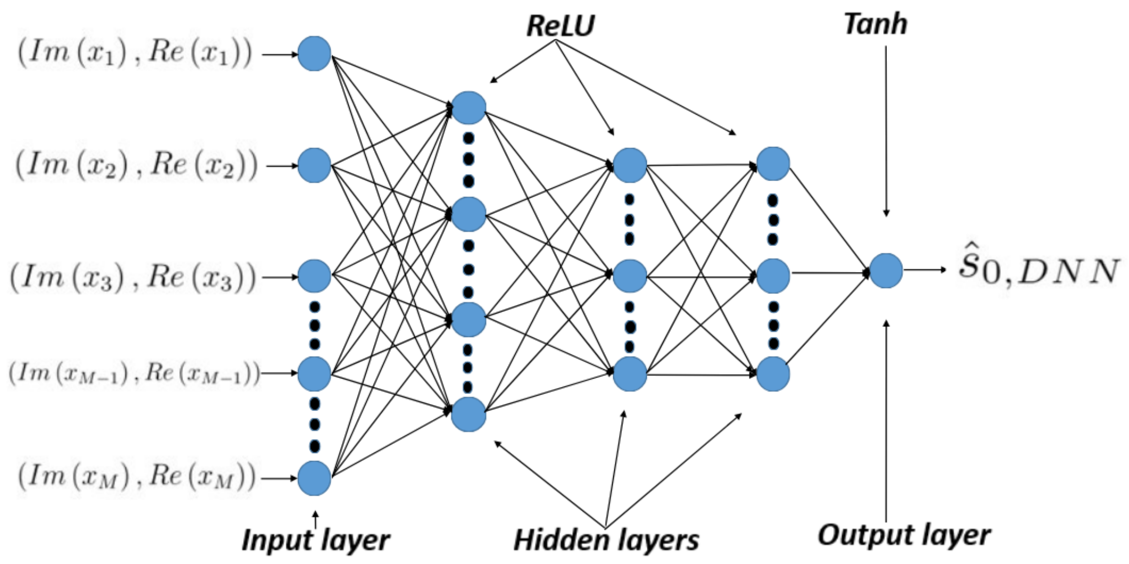


Figure 4.2: The architecture of the DNN.

4.4.2 Training Data Generation

We generate the data using the following steps: 1) We generate the BPSK symbols transmitted from the K transmitters $\{s_k^{(i)}\}$, $0 \leq k \leq K$. The superscript i denotes the index of the training sample. 2) We generate the channel realizations $\{\boldsymbol{h}_k^{(i)}\}$ and the AWGN $\{\boldsymbol{n}^{(i)}\}$. 3) For each tuple $\left(\{s_k^{(i)}\}, \{\boldsymbol{h}_k^{(i)}\}, \{\boldsymbol{n}^{(i)}\}\right)$, we generate the received signal $\{\boldsymbol{x}^{(i)}\}$ according to (4.1), and we get the estimates of the desired transmitter's BPSK symbols $\{\hat{s}_{0,PZF}^{(i)}\}$ by running the PZF algorithm using (4.2). The tuple $\left(\{Im\left(\boldsymbol{x}^{(i)}\right), Re\left(\boldsymbol{x}^{(i)}\right)\}, \{\hat{s}_{0,PZF}^{(i)}\}\right)$ represents the i^{th} training sample, where $Im\left(\cdot\right)$ and $Re\left(\cdot\right)$ represent the imaginary and the real part, respectively. 4) To obtain the entire training data, we repeat the above process multiple times. We use the same process to generate the validation data set which is used to perform cross validation and model selection. It is used to minimize overfitting by stopping the NN during the training stage.

4.4.3 Training stage

To optimize the weights of the NN, we use the entire training data set $\left(\{Im\left(\boldsymbol{x}^{(i)}\right), Re\left(\boldsymbol{x}^{(i)}\right)\}, \{\hat{s}_{0,PZF}^{(i)}\}\right)_{i\in\mathcal{T}}$, where $\mathcal{T}$ contains the indices for the training sets. We use the quadratic cost function which is the mean square error between the true label $\{s_{0,PZF}^{(i)}\}$ and the output of the network $\{\hat{s}_{0,DNN}^{(i)}\}$. The optimization algorithm we use for our NN is a mini-batch SGD.

4.4.4 Testing stage

In this stage, we generate the testing data using the same process used to generate the training data sets. Then, we pass each received signal $\{x^{(i)}\}$ through the trained network and obtain the estimates of the BPSK symbols $\{\hat{s}_{0,DNN}^{(i)}\}$. Finally, we compute the bit error probability (BEP) of the DNN and compare it with that obtained by the PZF.

4.5 Numerical Results

In Fig. 4.3, we study the impact of varying the batch size on the mean square error (MSE) evaluated on the validation set. As can be seen in Fig. 4.3, larger batch size leads to slower convergence and higher validation error. Based on this figure, we choose the batch size to be 200.

Figure 4.4 illustrates the MSE of the validation set during the learning process for different learning rates. The figure shows that by increasing the learning rate, the MSE converges to zero very fast. Therefore, we choose the learning rate to be 0.01.

Figure 4.5 shows the MSE versus the number of iterations at $\frac{E_b}{N_0} = 15$ dB during the training phase. As illustrated in Fig. 4.5, the MSE decreases sharply at first and then gradually declines until the MSE approaches zero during the training process of the network. The MSE in every Epoch was approximately the same, for training and validation data set, which means that the NN fits accurately the training data set.

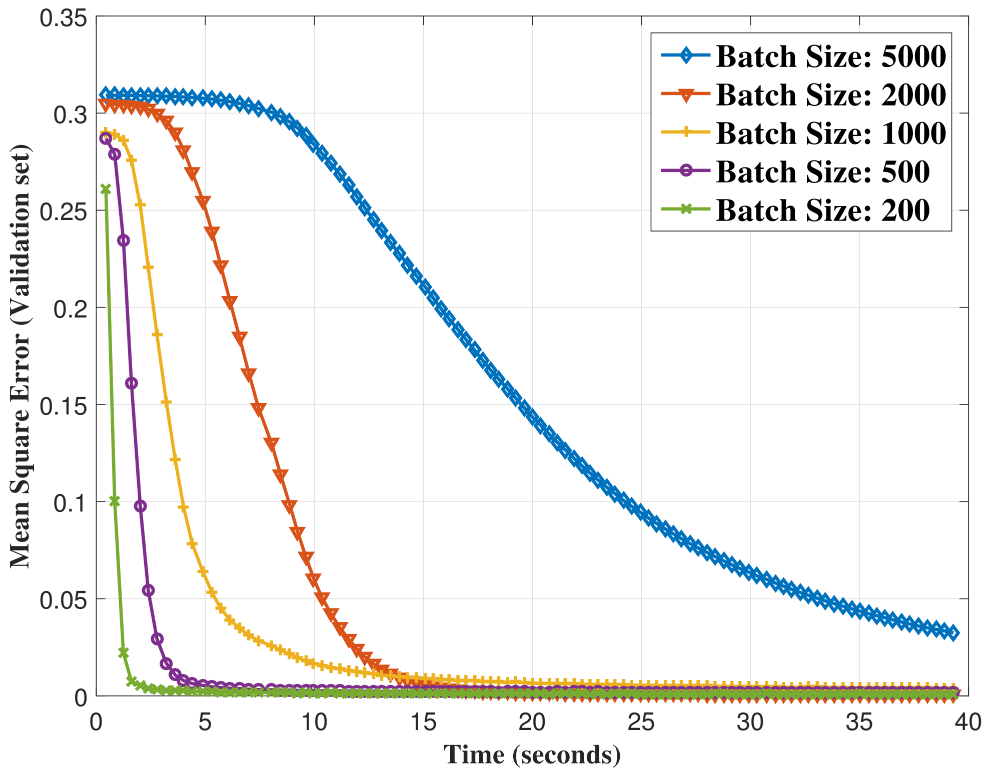


Figure 4.3: The MSE of the DNN during the learning process for different batch sizes at $\frac{E_b}{N_0} = 0$ dB. The network comprises 11 users and a receiver with 12 antennas. The transmit signal-to-noise-ratios (SNRs) of the users are equals to $\frac{E_b}{N_0}$ $\left(\frac{E_{b,i}}{N_0} = \frac{E_b}{N_0},\ 0 \leq i \leq K\right)$. N_c is equal to 10. We set μ_s =0 and σ_s =0 dB.

We studied the average BEP of the model-based and the learning-based PZF for 8 phase shift keying (8-PSK) and BPSK as shown in Fig. 4.6. To train the NN on 8-PSK and BPSK symbols, we used the same NN's architecture, i.e., the input, the first hidden, and the second hidden layers contain the same number of neurons for both BPSK and 8-PSK symbols, respectively. Since the outputs of the NN are the estimates of the desired transmitter's BPSK and 8-PSK symbols, the NN's output layer, when BPSK (8-PSK) respectively, symbols are used for transmission, contains one neuron (three neurons) respectively, because we can represent one BPSK (8-PSK) respectively, symbol as one bit (three bits) respectively. It can be seen in Fig. 4.6 that the error performance of the learning-based PZF is very close to the model-based PZF at low SNRs for both BPSK and 8-PSK symbols. For instance, the

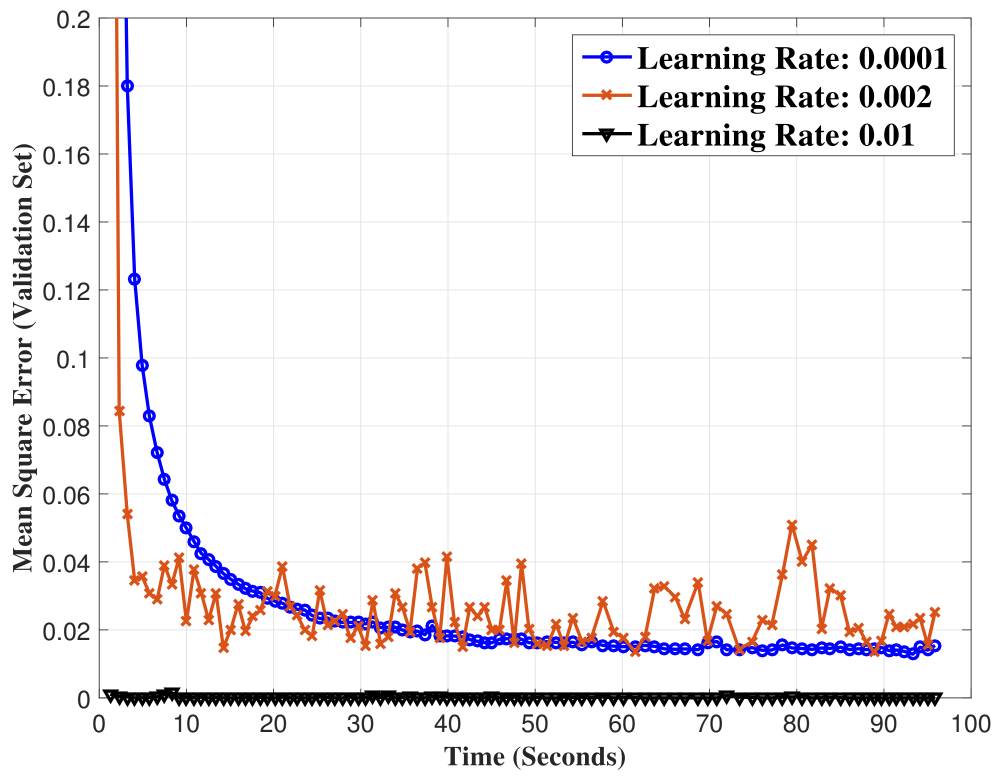


Figure 4.4: The MSE of the DNN during the learning process for different learning rates at $\frac{E_b}{N_0} = 0$ dB. The network comprises 11 users and a receiver with 12 antennas. The transmit SNRs of the users are equals to $\frac{E_b}{N_0}$ $\left(\frac{E_{b,i}}{N_0} = \frac{E_b}{N_0},\ 0 \leq i \leq K\right)$. N_c is equal to 10. We set μ_s =0 and σ_s =0 dB.

relative error between the model-based and the learning-based PZF performances, for both BPSK and 8-PSK is at most 6% at high SNR. However, learning-based PZF requires only a small fraction of the computational resources used by model-based PZF.

Table 4.1 illustrates the computation reduction which is defined as $\frac{t_{PZF}-t_{DNN}}{t_{PZF}}$, where t_{PZF} and t_{DNN} represent the total central processing unit (CPU) execution time in seconds for the model-based and the learning-based PZF, respectively, for different number of users K. It is observed that using learning-based instead of model-based PZF reduces substantially the implementation complexity in terms of computational time by more than 90%.

We studied the BEP of the model-based PZF and the learning-based PZF when MSE and cross entropy are used as loss functions for BPSK symbols as shown in Fig. 4.7. It can

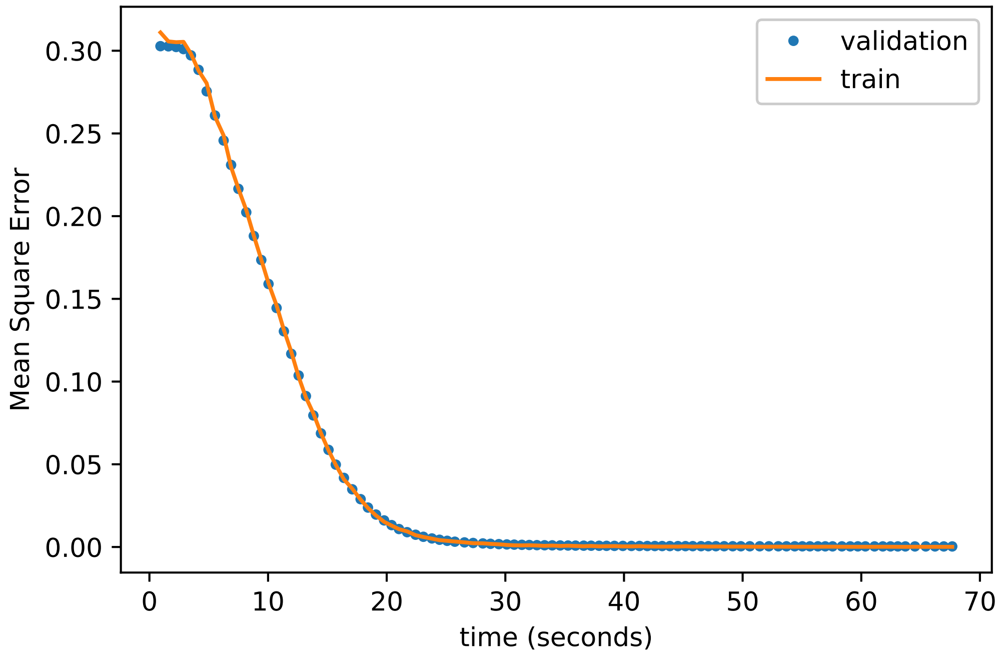


Figure 4.5: The MSE of the DNN during the learning process at $\frac{E_b}{N_0} = 15$ dB. The network comprises 11 users and a receiver with 12 antennas. The transmit SNRs of the users are equals to $\frac{E_b}{N_0}$ $\left(\frac{E_{b,i}}{N_0} = \frac{E_b}{N_0},\ 0 \leq i \leq K\right)$. N_c is equal to 10. We set μ_s =0 and σ_s =0 dB.

be seen in Fig. 4.7 that the error performance of learning-based PZF with MSE loss function matches the error performance of learning-based PZF with cross entropy loss function when $-5 \leq \frac{E_b}{N_0} < 20$. Furthermore, the relative error between the learning-based PZF with MSE loss function and the learning-based PZF with cross entropy loss function performances is at most 6% at high SNR, i.e., $20 \leq \frac{E_b}{N_0} \leq 25$. In addition, the BEPs of both leaning-based PZFs with MSE and cross entropy loss functions are very close to the model-based PZF's

Table 4.1: Computational complexity comparison for different number of users K. The receiver comprises 12 antennas. The transmit SNRs of the users are equals to $\frac{E_b}{N_0}$ $\left(\frac{E_{b,i}}{N_0} = 5 \text{ dB},\ 0 \leq i \leq K\right)$. N_c is equal to 10. We set μ_s =0 and σ_s =0 dB.

# of users (K)	t_{DNN} (sec.)	t_{PZF} (sec.)	$\frac{t_{PZF}-t_{DNN}}{t_{PZF}}$
11	0.047	0.851	94.48%
15	0.093	1.938	95.2%
19	0.149	5.103	97.08%

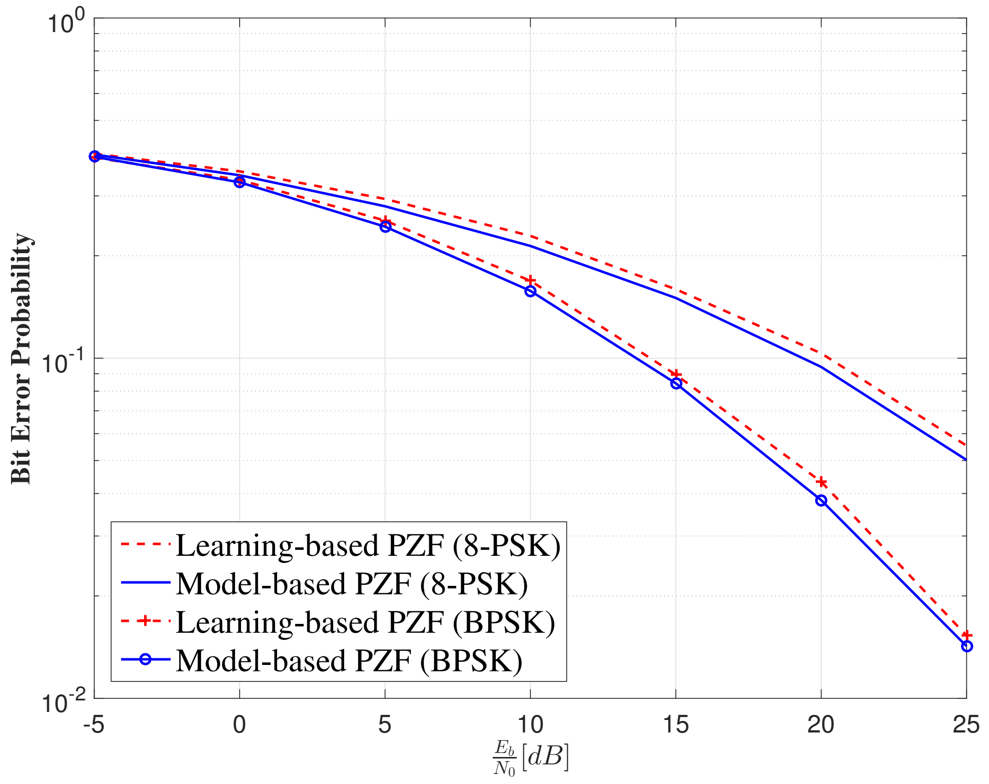


Figure 4.6: The BEP of model-based and learning-based PZF versus $\frac{E_b}{N_0}$ for BPSK and 8-PSK. The network comprises 11 users and a receiver with 12 antennas. The transmit SNRs of the users are equals to $\frac{E_b}{N_0}$ $\left(\frac{E_{b,i}}{N_0} = \frac{E_b}{N_0},\ 0 \le i \le K\right)$. N_c is equal to 10. We set μ_s =0 and σ_s =0 dB.

BEP.

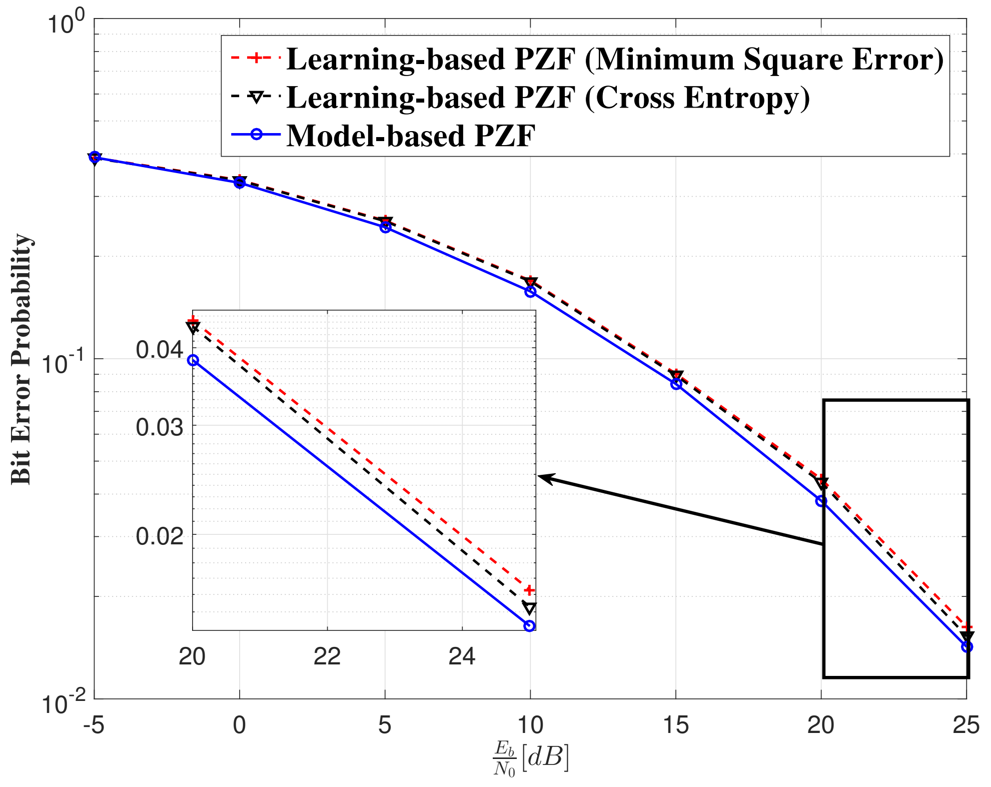


Figure 4.7: The BEP of model-based and learning-based PZF versus $\frac{E_b}{N_0}$ for BPSK symbols when MSE and cross entropy are used as loss functions. The network comprises 11 users and a receiver with 12 antennas. The transmit SNRs of the users are equals to $\frac{E_b}{N_0}$ $\left(\frac{E_{b,i}}{N_0} = \frac{E_b}{N_0},\ 0 \leq i \leq K\right)$. N_c is equal to 10. We set μ_s =0 and σ_s =0 dB.

CHAPTER 5

Symbol Detection and Channel Estimation Using Neural Networks in Optical Communication Systems

5.1 Motivation and Related Work

Studying communication performance limits, such as the channel capacity and the bit error rate, of the OWC system is more difficult than those of the RF systems. First, in OWC systems, the transmitted signal must satisfy nonnegativity, peak, and average intensity constraints due to practical and safety reasons. Second, a clear characterization of the optical channel model is not available in many OWC scenarios. Hence, designing communication systems that can efficiently perform without heavily relying on the channel model is much more preferable in OWC.

Conventionally, a ML rule has been used to detect transmitted symbols at the optical receiver [50]. But, this rule is deficient in two ways. First, it assumes the receiver knows that the channel follows a certain model, which leads to inaccurate symbol detection when the actual channel is different from the assumed model. Second, although detection algorithms based on the ML rule are optimal in terms of error performance, they usually entail a high computational complexity due to searching over all possible outcomes [1, 51]. Motivated by the success of deep learning in image recognition, natural language processing and recommender systems [14], and their ability to capture communication problems [42, 45, 52, 2], we propose a deep learning architecture for detection and estimation for OWC that does not rely on a channel model.

Recently, several studies suggested that learning-based communications algorithms can perform as well as their model-based counterparts. For example, the work in [42] proposes a NN architecture that implicitly estimates the CSI and detects symbols in an OFDM system. Two major problems are associated with this approach. First, CSI is not actually estimated but is assumed to be implicitly estimated by the NNs; hence, communications systems that require CSI for modulation classification and power control, e.g., [53, 54, 55], will have to obtain CSI using some other means. Second, while the proposed architecture provides a good detection accuracy, a study about the complexity of their proposed architecture during online implementation is not provided. The authors in [56] propose a learning-based approach relying on NNs to design a minimum mean square error (MMSE) channel estimator in uplink wireless communications. Simulation results show that the normalized mean square error performance is close to the performance of the MMSE channel estimator with low complexity execution. In [57], the authors proposed learning algorithms for coordinated beamforming for highly-mobile mm-Wave systems. Additionally, authors in [56] train NNs based on the assumption that the covariance matrices of the channel and the noise are known, i.e., the statistics of the channel and the noise are known, which might be impractical in real scenarios. Moreover, [58] shows that using pseudo-random bit sequences or short repeated sequences can severely degrade NNs' accuracy in optical communication systems because they can overfit and are, hence, biased towards predicting these sequences.

5.2 System Model and Background

We consider an OWC system, shown in Fig. 5.1, where a transmitter wishes to reliably communicate the message $m \in \mathcal{M}$, $\mathcal{M} = \{1, 2, \ldots, M\}$ over an optical channel to a receiver. To do so, m is first mapped into the sequence $\boldsymbol{X}^L \in \Omega^L$, where Ω is an optical modulation set. Now, the elements of $\boldsymbol{X}^L$ given by $X(i)$, $1 \leq i \leq L$ satisfy the nonnegativity and peak intensity constraints dictated by the physical properties of the optical channel, i.e., $0 \leq X(i) \leq A$, [59]. Let $T = LT_s$ be the coherence time of the optical channel, where L is the number of channel uses and T_s is the symbol duration. The communication rate of this OWC system is $\frac{k}{L}$ bits/channel use, where $k = \log_2(M)$ bits are transmitted through L channel uses. Afterwards, the sequence $\boldsymbol{X}^L$ is transmitted through an optical wireless channel and the sequence $\boldsymbol{Y}^L \in \mathbb{R}^L$ is received by the optical receiver. Here, a specific channel model is not considered and the received sequence $\boldsymbol{Y}^L$ can be generated according to the probabilistic law given by

$$\mathcal{P}\left(\boldsymbol{Y}^L \,|\, \boldsymbol{X}^L, h\right), \tag{5.1}$$

where $h \in \mathbb{R}_+$ represents the optical channel gain and is assumed to be constant during the transmission of the sequence $\boldsymbol{X}^L$. Furthermore, the positivity of h follows from the fact that in OWC, information is communicated through the intensity of the light [50]. Note that the channel transition probability law given by (5.1) need not be known for the proposed approach to work. As we argue below, such a law will be learned by the NN during the training stage. Furthermore, we consider a pilot-based channel estimation approach, in which the pilot symbol X_p is used for channel estimation and is transmitted as the first symbol $X(1)$ of the transmitted sequence, i.e., $X_p \triangleq X(1)$. In the following section, we describe the proposed NN-based estimation and detection algorithms.

5.3 Proposed Learning-Based Receiver Design for Channel Estimation and Detection

As mentioned before, the ML detection algorithm requires the knowledge of the channel model and even when the channel model is available, ML entails a large computation complexity. Therefore, we propose a learning-based receiver design that does not rely on a channel model, can estimate CSI, and can achieve the optimum performance of the model-based ML detector. Figure 5.1 shows a typical communication system incorporating the proposed learning-based channel estimator and neural detector.

5.3.1 Proposed Learning-Based Estimation Scheme

In this section, we propose the NN-based estimator that outputs $\hat{h}$ as an estimate of h.

The NN Architecture: The neural estimator NN($\hat{h}$) consists of one hidden layer, a sigmoid activation function at the hidden layer, and a linear activation function at the output layer. The input to NN($\hat{h}$) is the first element of the received sequence $Y(1)$, and the output

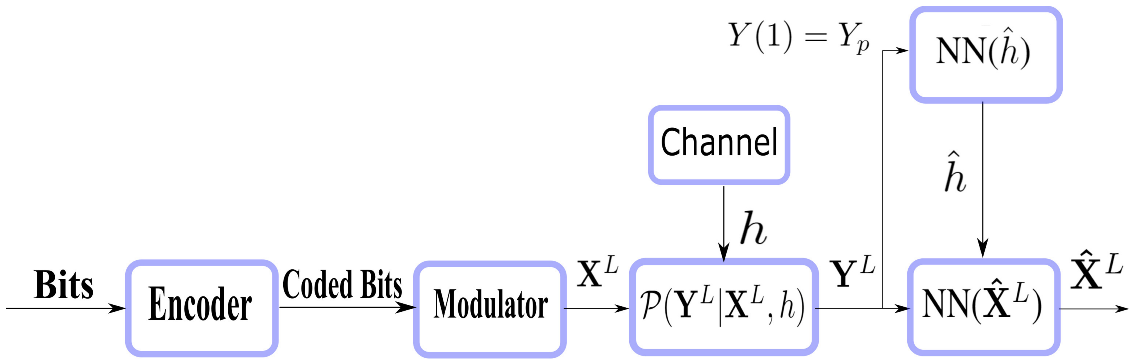


Figure 5.1: The implementation of the proposed NN-based channel estimation and symbols detection (at the receiver) in a communication system. The transmitter encodes a stream of bits into coded bits using appropriate coding schemes. Then, a modulator maps the coded bits into the sequence $\mathbf{X}^L \in \Omega^L$ with the first symbol $X(1) = X_p$ fixed as a pilot, which passes over an optical channel. At the receiver, the NN-based channel estimator NN($\hat{h}$) uses the first element $Y(1) = Y_p$ of the received sequence $\mathbf{Y}^L \in \mathbb{R}^L$ to obtain an estimate of the channel gain $\hat{h}$. Then, the NN-based detector NN($\hat{\mathbf{X}}^L$) uses $\hat{h}$ and the received sequence $\mathbf{Y}^L$ to estimate the transmitted symbols as $\hat{\mathbf{X}}^{\mathbf{L}} \in \Omega^L$.

is the estimated CSI, $\hat{h}$.

Training Data Generation: The training data for channel estimation is generated according to the following steps:

- We first generate the channel coefficients $\{h^n\}$ with $1 \leq n \leq N_s$, where N_s is the number of training samples.
- We then generate the first element of the received signals $\{Y_p^n\}$ according to (5.1) for the n^{th} training sample.
- Finally, we label the training data as the tuple $\{Y_p^n, h^n\}$.

Training stage: The loss function $\mathcal{L}(\hat{h}, h)$ is the normalized mean square error (NMSE), defined as:

$$\mathcal{L}(\hat{h}, h) = \frac{1}{N_s} \sum_{n=1}^{N_s} (\hat{h}^n - h^n)^2. \tag{5.2}$$

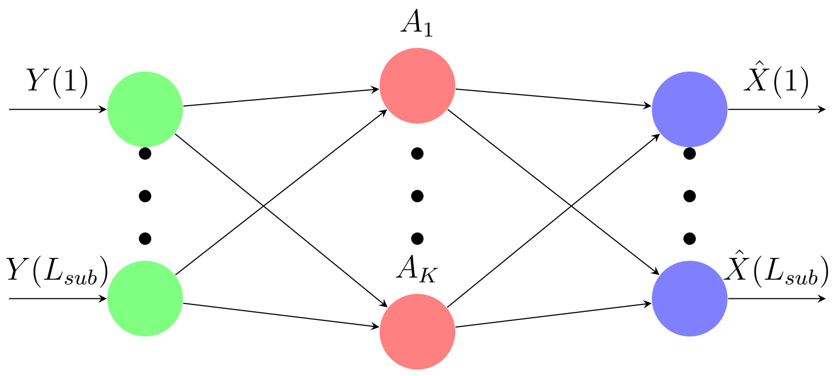
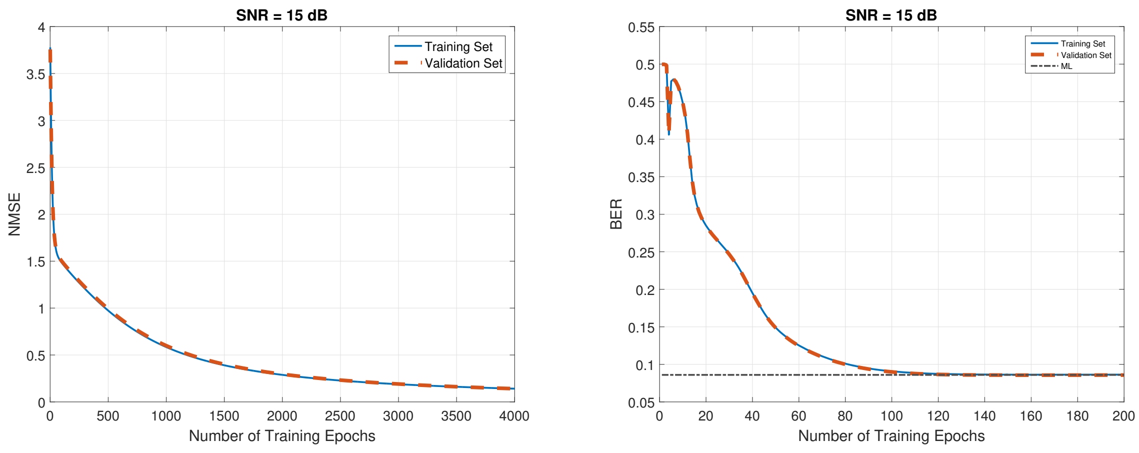


Figure 5.2: The NN used for detection. A sigmoid activation function is used for the hidden and output layers, where A_i denotes the ith output at the hidden layer.



(a) NMSE for NNs-based channel estimator during the training phase.

(b) BER for NNs-based detector during the training phase.

Figure 5.3: Error performance of NNs-based channel estimator and detector as the number of training epochs increases, given a training SNR of 15 dB.

Then, the task during the training stage can be formulated as:

$$\underset{\hat{h}}{\text{minimize}} \quad \mathcal{L}(\hat{h}, h), \tag{5.3a}$$

$$\text{subject to} \quad 0 \leq \hat{h} < \infty. \tag{5.3b}$$

To solve the optimization problem in (5.3), the gradient decent algorithm is implemented with backpropagation to update the weights at the hidden and output layers as in [60, eq. (4.4)–(4.11)].

Testing stage: In the testing stage, new data is generated to examine the performance of the NN estimator. The data is generated as in the aforementioned steps for generating the training data. The NN-based estimator uses the received signal $Y_p \triangleq Y(1)$ to obtain an estimate of the channel $\hat{h}$.

5.3.2 Proposed Learning-Based Detection Scheme

We now present a learning-based detection scheme that uses NNs to detect the transmitted symbols sequence $\boldsymbol{X}^L$.

The NN Architecture: Figure 5.2 illustrates the structure of the NN used for detection. The received signals $Y\left(j\right)$, $1 \leq j \leq L_{sub}$, where $L_{sub} \leq L$, are the inputs to the network. Hence, the input layer contains L_{sub} neurons. L_{sub} represents the length of the subsequence that it is contained in the whole sequence of length L. In this case, $L = qL_{sub}$, with q $\in \mathbb{N}$. The outputs of the network are the estimates of the transmitter's symbols via NN, i.e., $\hat{X}\left(j\right)$, $1 \leq j \leq L_{sub}$. Therefore, the output layer contains L_{sub} neurons. The hidden layer contains $K = 2L_{sub}$ neurons. A sigmoid function is used as an activation function for the hidden and output layers. In this way, L_{sub} symbols are detected through each pass through the NN. The process is then repeated until all L symbols are detected. If current received signal is uncorrelated with previous ones, it is enough to use $L_{sub} = 1$. But if the received signals are correlated in time, then we recommend using L_{sub} equal to the channel memory length. Clearly, as L_{sub} increases, the training complexity (in terms of the number of required operations) increases.

Training Data Generation: We generate the training data using the following steps:

- We first generate the transmitted sequences $\{\boldsymbol{X}^{L,n}\}$ and the channel coefficients $\{h^n\}$, where the superscript n denotes the index of the training sample.
- We then generate the received signals $\{\boldsymbol{Y}^{L,n}\}$ according to (5.1).
- The tuple $\left(\{\boldsymbol{Y}^{L,n}\}, \{\boldsymbol{X}^{L,n}\}\right)$ represents the n^{th} training sample.

Training Stage: We use the entire training data set $(\{\boldsymbol{Y}^{L,n}\}, \{\boldsymbol{X}^{L,n}\})_{n\in\mathcal{T}}$, where $\mathcal{T}$ contains the indices for the training sets. Furthermore, we employ the quadratic loss function $\mathcal{L}(\hat{\mathbf{X}}, \mathbf{X})$ which is defined as the mean square error between the true label $\{\boldsymbol{X}^{L,n}\}$ and the output of the network $\{\hat{\boldsymbol{X}}^{L,n}\}$ and is given by

$$\mathcal{L}(\hat{\mathbf{X}}^L, \mathbf{X}^L) = \frac{1}{N_s L} \sum_{n=1}^{N_s} \|\hat{\mathbf{X}}^{L,n} - \mathbf{X}^{L,n}\|^2, \tag{5.4}$$

where $\|\cdot\|$ is the Euclidean norm. Therefore, the optimization problem can be formalized as:

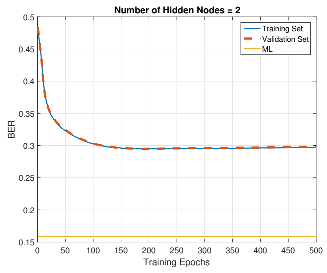
$$\begin{aligned}
&\underset{\hat{\mathbf{X}}^L}{\text{minimize}} && \mathcal{L}(\hat{\mathbf{X}}^L, \mathbf{X}^L) && \text{(5.5a)}\\
&\text{subject to} && 0 \leq \hat{X}(i) \leq A, && \text{(5.5b)}\\
& && \forall\, i = 1, \ldots, L_{sub}. && \text{(5.5c)}
\end{aligned}$$

To solve the problem in (5.5), the gradient decent is used with backpropagation.

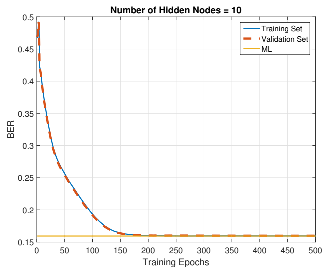
Testing Stage: We generate the testing data using the same process used to generate the training data, except that we use the estimated channels $\hat{h}^n$ obtained by NN$(\hat{h}^n)$. Then, we pass each received signal $\boldsymbol{Y}^L$ through the trained network NN$(\hat{\boldsymbol{X}}^L)$ and obtain the estimated sequence $\hat{\boldsymbol{X}}^L$.

5.4 Numerical Results

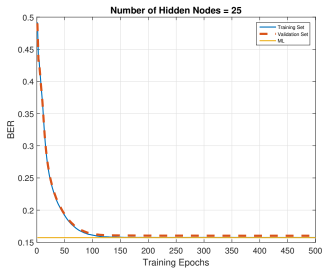
In this section, for the sake of training data generation for NN-based estimation and detection schemes, we consider that the received sequence $\boldsymbol{Y}^L$ is generated by a specific model. The considered model also allows for performance comparison between the learning-based methods and a model-based algorithm benchmark, such as ML. In particular, we



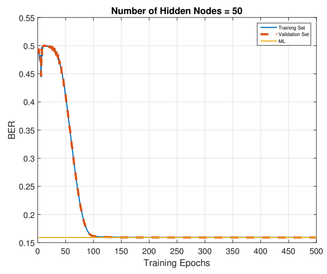
(a) $K = 2$.



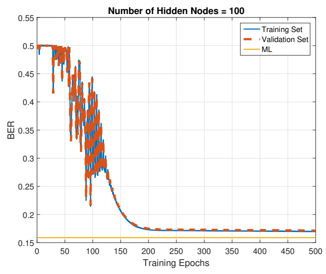
(b) $K = 10$.



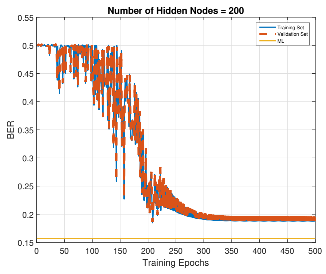
(c) $K = 25$.



(d) $K = 50$.



(e) $K = 100$.



(f) $K = 200$.

Figure 5.4: The effect of changing the number of hidden nodes, K, on the training error.

consider that the elements of the received sequence $\boldsymbol{Y}^L$ are generated by

$$Y(i) = hX(i) + Z(i), \quad \text{for } 1 \leq i \leq L, \tag{5.6}$$

where h is distributed according to the Lognormal distribution with variance 1, which is constant during L channel uses, and $Z(i) \sim \mathcal{N}(0, \sigma^2)$ is the AWGN. Note that the Lognormal distribution is an accurate model for a free space optical link operating over weak turbulence channels [50, Ch. 9]. Assuming the channel model in (5.6), the simulated training and testing data sets are generated according to the steps described in Sec. III-B.

Next, we briefly review the ML detection algorithm. The ML detector searches over the space of all possible symbols in the modulation set Ω to find the symbol $\hat{X}(i)$ that maximize the likelihood of obtaining the received sequence [1], i.e.,

$$\hat{X}(i) = \underset{X(j)\in\Omega,\ j=1,\ldots,L}{\arg\max} \ \mathcal{P}(Y(i)|X(j)) \quad \text{for } 1 \leq i \leq L. \tag{5.7}$$

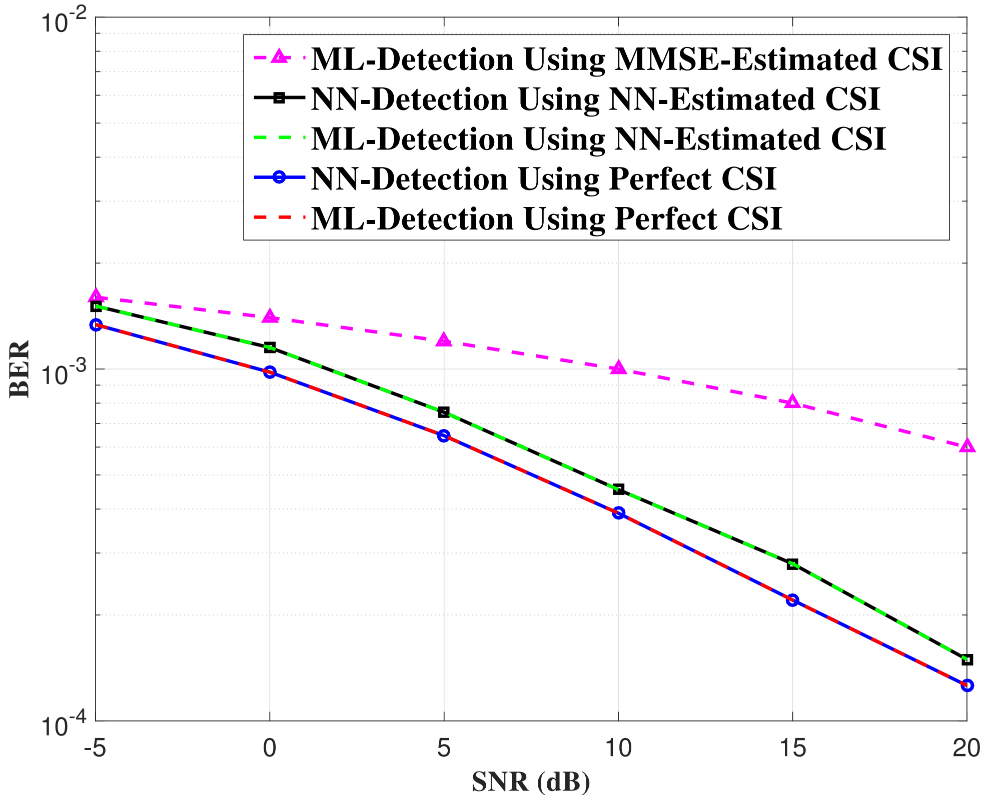


Figure 5.5: Comparison of BER versus SNR performance between a NN-based detector and a ML detector under different CSI assumptions, namely: under CSI obtained using a NN-based and MMSE channel estimators and under perfect CSI (at the receiver).

Since the noise $Z(i)$ is Gaussian, then (5.7) is equivalent to:

$$\hat{X}(i) = \underset{X(j)\in\Omega,\ j=1,\ldots,L}{\arg\max} \left|Y(i) - hX(j)\right| \quad \text{for } 1 \leq i \leq L. \tag{5.8}$$

We now illustrate the performance of the proposed scheme to estimate the channel and then detect transmitted symbols at the receiver. We assume that the OWC system employs ON-OFF keying (OOK) modulations (a modulation scheme often used in OWC systems because it satisfies the nonnegativity and the peak intensity constraints dictated by the physical properties of the optical channel [59, 61, 62]). The numerical results herein assume $L = 128$, and that a pilot sequence of length one is used out of the 128 symbols for channel estimation. The pilot sequence that has been corrupted by fading and AWGN is fed to the

NN-based channel estimator. The remaining sequence of random signals (of length 127) and the output of the channel estimator (i.e., $\hat{h}$) are fed to the NN-based detector, whose output is an estimate of the transmitted symbols (i.e., $\hat{\mathbf{X}}$). The size of each of the training, validation, and testing data sets is 20000, 20000, and 10000 sample sequences, respectively. Each dataset is randomly generated according to the steps outlined in Section 5.3. The validation dataset is used to illustrate the performance of the NN on unseen data.

Figure 5.3 shows the training errors for the NN-based channel estimator and the NN-based detector, respectively. In Fig. 5.3(a), the NMSE between the true CSI and the estimated CSI approaches zero as the number of training epochs increases. This trend happens for both the training data and the validation data, which means that the NN architecture designed herein avoids over-training and is also able to achieve low estimation and detection errors on new data. In Fig. 5.3(b), the bit error rate (BER) performance of NN-based detector approaches that of the ML detector, which is a lower bound on detection error but entails a high computational complexity and requires a channel model.

To demonstrate that setting $K = 2L_{sub}$ provides optimal detection accuracy while maintaining a low implementation complexity, we simulate the effect of changing K on the training error during the detection stage for a fixed L_{sub}. Figure 5.4 shows several attempts for obtaining a good accuracy during the training stage by varying the number of hidden nodes K when $L_{sub} = 5$. Note that the SNR in Fig. 5.4 is 0 dB. Figure 5.4(a) shows that having only two hidden nodes leads to achieving an error floor that is much greater that the lower bound ML error. But as the number of hidden nodes increases to ten and twenty five in Figures 5.4(b) and 5.4(c), respectively, the NNs BER performance gets closer to the BER performance of the optimal ML detector. When the number of hidden nodes increases to fifty, one hundred, and two hundred as shown in Figures 5.4(d)-5.4(f), the BER changes rapidly between iterations. We stipulate that these rapid changes in the BER can be ameliorated by varying the learning rates when the number of hidden nodes increases. But since computational complexity is important and since ten hidden nodes achieve an accuracy comparable to that of the ML detector, it is sufficient to use only ten hidden nodes in this case. We also observed this pattern for numerous values of L_{sub}. Hence, the experiments suggest that using $K = 2L_{sub}$ achieves high detection accuracy while maintaining an acceptable implementation complexity.

To examine the robustness and accuracy of the proposed NN-based channel estimator and detector, we simulate the error performance of the proposed method during the online phase for various SNRs. Figure 5.5 compares the BER versus SNR performance of the proposed NN-based channel estimator and symbols detector with the ML along with MMSE channel estimator and with the ML detector along with perfect CSI. We observe that the BER performance of the NN-based detector closely matches the BER performance of the ML detector across all SNRs, under different CSI assumptions. In addition, the NN-based detector with NN-based channel estimator provides 14 dB gain compared to the ML-based detector with MMSE channel estimator at BER of 6×10^{-4}. For the case of the perfect CSI at the receiver, the NN-based detector achieves the optimal performance of the ML detector across all SNRs. Furthermore, when the NN-based channel estimator is employed to obtain the CSI rather than having perfect CSI, the error performance shifts by less than 1 dB, for the BERs that are less than 10^{-3}. In this case, both the NN-based detector and the ML detector assume that the estimated CSI from the NNs based estimator is the true CSI.

CHAPTER 6

Concluding Remarks

6.1 Summary

In this thesis, we investigated the design of learning-based algorithms for the wireless communication systems.

In chapter 3, we derived an upper bound on the expected generalization error of the CNN in terms of the entropy of the input and the number of layers. The upper bound shows that when there is no one-to-one mapping between hidden layers, the expected generalization error decreases to zero as the number of hidden layers L increases in the CNN. It also shows that adding more layers is not rewarding when there is at least one one-to-one mapping between two consecutive hidden layers.

In chapter 4, we designed a new DNN-based scheme for PIC to overcome the computation complexity of PZF. Our empirical results indicate that NNs can be trained to well-mimic the behaviour of PZF. In many aspects, we expect that DNNs could be used in many computationally expensive signal processing tasks with stringent real-time requirements.

Finally, in chapter 5, we proposed and designed a novel NN-based scheme for channel estimation and symbol detection for OWC systems that does not require a channel model. The empirical results indicate that NNs can be trained to accurately estimate the CSI and closely mimic the behavior of the ML detector. We observed that the proposed NN-based scheme using two architectures for channel estimation and symbols detection, each containing only one hidden layer for reduced complexity, provides comparable estimation and detection accuracy to that of the model-based ML algorithm.

6.2 Future Research

The work presented in this thesis can be extended in different ways. First, it will be interesting to find a lower bound on the expected generalization error which depends on the CNN's parameters such as the number of training samples and the number of layers. Second, extending the proposed scheme in chapter 4 to multiple-input-multiple-output (MIMO) PIC system, and to see how NNs perform under different wireless channels models such as Nakagami channel model is also of interest. Finally, one could extend the proposed learning-based scheme in chapter 5 to include power control and modulation classification in OWC system, and to see how NNs perform under different optical channels models, such as input-dependent and Poisson channels.

Bibliography

[1] X. Zhu and R. D. Murch, "Performance analysis of maximum likelihood detection in a MIMO antenna system," *IEEE Transactions on Communications*, vol. 50, no. 2, pp. 187–191, Feb 2002.

[2] 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, Dec 2017.

[3] W. Fatnassi and Z. Rezki, "Reliability Enhancement of Smart Metering System Using Millimeter Wave Technology," *IEEE Transactions on Communications*, vol. 66, no. 10, pp. 4877–4892, Oct 2018.

[4] N. Farsad and A. Goldsmith, "Neural network detectors for sequence detection in communication systems," *IEEE Transactions on Signal Processing*, 2018.

[5] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, "Millimeter Wave Mobile Communications for 5G Cellular: It Will Work!" *IEEE Access*, vol. 1, pp. 335–349, 2013.

[6] V. Nurmela, A. Karttunen, A. Roivainen, L. Raschkowski, V. Hovinen, J. Y. EB, N. Omaki, K. Kusume, A. Hekkala, R. Weiler *et al.*, "Deliverable d1. 4 METIS channel models," in *Proc. Mobile Wireless Commun. Enablers Inf. Soc.(METIS)*, 2015, p. 1.

[7] M. K. Samimi and T. S. Rappaport, "3-d millimeter-wave statistical channel model for 5g wireless system design," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 7, pp. 2207–2225, July 2016.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[9] U. Challita, L. Dong, and W. Saad, "Proactive resource management in LTE-U systems: A deep learning perspective," *arXiv preprint arXiv:1702.07031*, 2017.

[10] R. C. Daniels, C. M. Caramanis, and R. W. Heath, "Adaptation in Convolutionally Coded MIMO-OFDM Wireless Systems Through Supervised Learning and SNR Ordering," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 114–126, Jan 2010.

[11] S. K. Pulliyakode and S. Kalyani, "Reinforcement learning techniques for Outer Loop Link Adaptation in 4G/5G systems," *arXiv preprint arXiv:1708.00994*, 2017.

[12] J. Vieira, E. Leitinger, M. Sarajlic, X. Li, and F. Tufvesson, "Deep convolutional neural networks for massive MIMO fingerprint-based positioning," in *28th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, PIMRC 2017, Montreal, QC, Canada, October 8-13, 2017*, 2017, pp. 1–6.

[13] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

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

[15] M. A. Nielsen, *Neural networks and deep learning.* Determination press San Francisco, CA, USA:, 2015, vol. 25.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[17] Y. Bengio, "Learning Deep Architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, Jan. 2009.

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

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[20] "Quickdraw dataset." [Online]. Available: https://github.com/googlecreativelab/quickdraw-dataset

[21] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," in *Advanced lectures on machine learning.* Springer, 2004, pp. 169–207.

[22] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[23] O. Bousquet and A. Elisseeff, "Stability and generalization," *Journal of machine learning research*, vol. 2, no. Mar, pp. 499–526, 2002.

[24] S. Shahrampour, A. Beirami, and V. Tarokh, "On Data-Dependent Random Features for Improved Generalization in Supervised Learning," in *2018 AAAI Conference on Artificial Intelligence*, 2018.

[25] V. Feldman and T. Steinke, "Calibrating Noise to Variance in Adaptive Data Analysis," in *COLT*, ser. Proceedings of Machine Learning Research, vol. 75, 2018, pp. 535–544.

[26] B. Neyshabur, S. Bhojanapalli, and N. Srebro, "A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks," in *International Conference on Learning Representations*, 2018.

[27] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 6240–6249.

[28] R. Bassily, S. Moran, I. Nachum, J. Shafer, and A. Yehudayoff, “Learners that Use Little Information,” in *ALT*, ser. Proceedings of Machine Learning Research, vol. 83, 2018, pp. 25–55.

[29] D. Russo and J. Zou, “Controlling Bias in Adaptive Data Analysis Using Information Theory,” in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, vol. 51, 09–11 May 2016, pp. 1232–1240.

[30] M. Raginsky, A. Rakhlin, M. Tsao, Y. Wu, and A. Xu, “Information-theoretic analysis of stability and bias of learning algorithms,” in *2016 IEEE Information Theory Workshop (ITW)*, Sep. 2016, pp. 26–30.

[31] J. Jiao, Y. Han, and T. Weissman, “Dependence measures bounding the exploration bias for general measurements,” in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 1475–1479.

[32] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[33] Y. Polyanskiy and Y. Wu, “A Note on the Strong Data-Processing Inequalities in Bayesian Networks,” *CoRR*, vol. abs/1508.06025, 2015.

[34] D. Jakubovitz, R. Giryes, and M. R. D. Rodrigues, “Generalization Error in Deep Learning,” *CoRR*, vol. abs/1808.01174, 2018.

[35] “Theano,” http://deeplearning.net/software/theano/.

[36] R. Shwartz-Ziv and N. Tishby, “Opening the Black Box of Deep Neural Networks via Information,” *CoRR*, vol. abs/1703.00810, 2017. [Online]. Available: http://arxiv.org/abs/1703.00810

[37] A. Kraskov, H. Stögbauer, and P. Grassberger, “Estimating mutual information,” *Phys. Rev. E*, vol. 69, p. 066138, Jun 2004.

[38] "Python Code for Entropy Estimator." [Online]. Available: {https://github.com/ravidziv/IDNNs/blob/master/idnns/information/entropy_estimators.py}

[39] "Non-Parametric Entropy Estimation Toolbox (NPEET)." [Online]. Available: https://www.isi.edu/~gregv/npeet.html

[40] "Python Code for Mutual Information Estimator." [Online]. Available: https://github.com/ravidziv/IDNNs/blob/master/idnns/information/mutual_info_estimation.py

[41] J. Schmidhuber, "Deep Learning in Neural Networks," *Neural Netw.*, vol. 61, no. C, pp. 85–117, Jan. 2015.

[42] H. Ye and G. Y. Li and B. H. Juang, "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, Feb 2018.

[43] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[44] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th annual international conference on machine learning.* ACM, 2009, pp. 873–880.

[45] H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," in *The International Zurich Seminar on Information and Communication (IZS 2018) Proceedings.* ETH Zurich, 2018, pp. 48–50.

[46] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep Learning Based Communication Over the Air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, Feb 2018.

[47] A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. t. Brink, "OFDM-Autoencoder for End-to-End Learning of Communications Systems," *arXiv preprint arXiv:1803.05815*, 2018.

[48] N. Jindal and J. G. Andrews and S. Weber, "Multi-Antenna Communication in Ad Hoc Networks: Achieving MIMO Gains with SIMO Transmission," *IEEE Transactions on Communications*, vol. 59, no. 2, pp. 529–540, February 2011.

[49] A. Goldsmith, *Wireless Communications.* Cambridge, U.K: Cambridge Univ. Press, 2005.

[50] M. Uysal, C. Capsoni, Z. Ghassemlooy, A. Boucouvalas, and E. Udvary, *Optical wireless communications: an emerging technology.* Springer, 2016.

[51] J. Jalden, B. Ottersten, and W.-K. Ma, "Reducing the average complexity of ML detection using semidefinite relaxation," in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 3, March 2005, pp. 1021–1024.

[52] E. Nachmani and E. Marciano and L. Lugosch and W. J. Gross and D. Burshtein and Y. Beéry, "Deep Learning Methods for Improved Decoding of Linear Codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, Feb 2018.

[53] M. Abu-Romoh, A. Aboutaleb, and Z. Rezki, "Automatic Modulation Classification Using Moments and Likelihood Maximization," *IEEE Communications Letters*, vol. 22, no. 5, pp. 938–941, May 2018.

[54] L. Han, F. Gao, Z. Li, and O. A. Dobre, "Low Complexity Automatic Modulation Classification Based on Order-Statistics," *IEEE Transactions on Wireless Communications*, vol. 16, no. 1, pp. 400–411, Jan 2017.

[55] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for wireless resource management," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, July 2017, pp. 1–6.

[56] D. Neumann, T. Wiese, and W. Utschick, "Learning the MMSE Channel Estimator," *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2905–2917, June 2018.

[57] A. Alkhateeb, S. Alex, P. Varkey, Y. Li, Q. Qu, and D. Tujkovic, "Deep Learning Coordinated Beamforming for Highly-Mobile Millimeter Wave Systems," *IEEE Access*, vol. 6, pp. 37 328–37 348, 2018.

[58] T. A. Eriksson, H. Bülow, and A. Leven, "Applying Neural Networks in Optical Communication Systems: Possible Pitfalls," *IEEE Photonics Technology Letters*, vol. 29, no. 23, pp. 2091–2094, Dec 2017.

[59] S. Arnon, J. Barry, G. Karagiannidis, R. Schober, and M. Uysal, *Advanced optical wireless communication systems.* Cambridge university press, 2012.

[60] S. Marsland, *Machine learning: an algorithmic perspective.* CRC press, 2015.

[61] H. Hemmati, *Deep space optical communications.* John Wiley & Sons, 2006, vol. 11.

[62] H. Hemmati, A. Biswas, and I. B. Djordjevic, "Deep-Space Optical Communications: Future Perspectives and Applications," *Proceedings of the IEEE*, vol. 99, no. 11, pp. 2020–2039, Nov 2011.

[63] U. von Luxburg and B. Schölkopf, *Statistical Learning Theory: Models, Concepts, and Results.* Amsterdam, Netherlands: Elsevier North Holland, May 2011, vol. 10, pp. 651–706.

Appendix A: Proof of Theorem 2: The Upper Bound on the Expected Generalization Error

Starting from (3.7), we have:

$$\begin{aligned}
& GEN\left(P_Z, P_{W|\boldsymbol{S}}\right) \\
= & \underset{P_{(\boldsymbol{S},W)}}{\mathbb{E}}\left[\underset{P_{\boldsymbol{S}'}}{\mathbb{E}}\left[\frac{1}{n}\sum_{i=1}^{n} l\left(W, Z_i'\right)\right] - \frac{1}{n}\sum_{i=1}^{n} l\left(W, Z_i\right)\right] && (6.1)\\
= & \underset{(P_{\boldsymbol{S}'},P_W)}{\mathbb{E}}\left[\frac{1}{n}\sum_{i=1}^{n} l\left(W, Z_i'\right)\right] - \underset{P_{(\boldsymbol{S},W)}}{\mathbb{E}}\left[\frac{1}{n}\sum_{i=1}^{n} l\left(W, Z_i\right)\right] && (6.2)\\
= & \underset{(P_{\boldsymbol{S}},P_W)}{\mathbb{E}}\left[\frac{1}{n}\sum_{i=1}^{n} l\left(W, Z_i\right)\right] - \underset{P_{(\boldsymbol{S},W)}}{\mathbb{E}}\left[\frac{1}{n}\sum_{i=1}^{n} l\left(W, Z_i\right)\right] && (6.3)\\
= & \underset{(P_Z,P_W)}{\mathbb{E}}\left[l\left(W, Z\right)\right] - \underset{P_{(Z,W)}}{\mathbb{E}}\left[l\left(W, Z\right)\right], && (6.4)
\end{aligned}$$

where (6.1) follows using the ghost sample principle [63] in which $\boldsymbol{S'} = [Z_1', \cdots, Z_n']$ has i.i.d components with distribution $P_{\boldsymbol{S}'} = P_{\boldsymbol{S}}$ and is independent from $\boldsymbol{S}$. Note that the first expectation in (6.4) is over the marginals P_Z and P_W, whereas the second expectation is over the joint $P_{(Z,W)}$. Next, we bound (6.4) as follows:

$$\begin{aligned}
& |\underset{(P_Z,P_W)}{\mathbb{E}}\left[l\left(W, Z\right)\right] - \underset{P_{(Z,W)}}{\mathbb{E}}\left[l\left(W, Z\right)\right]| \\
= & |\sum_{i,j}\left(P_Z\left(Z_i\right)P_W\left(W_j\right) - P_{(Z,W)}\left(Z_i, W_j\right)\right) l\left(W_j, Z_i\right)| && (6.5)\\
\leq & \sum_{i,j}|P_Z\left(Z_i\right)P_W\left(W_j\right) - P_{(Z,W)}\left(Z_i, W_j\right)||l\left(W_j, Z_i\right)| && (6.6)\\
\leq & \|P_Z P_W - P_{(Z,W)}\|_1 \|l\left(W, Z\right)\|_\infty && (6.7)\\
\leq & \|P_Z P_W - P_{(Z,W)}\|_1, && (6.8)
\end{aligned}$$

where the sum in (6.5) is a double sum over $i's$ and $j's$, $i, j = 1, \ldots, n$, and $\|\cdot\|_1$ and $\|\cdot\|_\infty$ represent the $L1$-norm and the infinity norm, respectively. Note that $\|l\left(W, Z\right)\|_\infty \leq 1$ since the loss function is assumed to be bounded. Now, we use the following Theorem which provides an upper bound on the $L1$-norm in (6.8):

Theorem 3. *[32] For any distributions P and Q, the following upper bound holds:*

$$\|P-Q\|_1^2 \leq 2\log(2)\, D(P\|Q), \tag{6.9}$$

where $D(P\|Q)$ is the Kullback-Leibler (KL)-divergence between P and Q.

Using Theorem 3, we have:

$$\begin{aligned} GEN\left(P_Z, P_{W|\boldsymbol{S}}\right) &= \left|\mathop{\mathbb{E}}_{(P_Z,P_W)}[l(W,Z)] - \mathop{\mathbb{E}}_{P_{(Z,W)}}[l(W,Z)]\right| \\ &\leq \sqrt{2\log(2)\, D\left(P_Z P_W \| P_{(Z,W)}\right)} \qquad (6.10) \\ &= \sqrt{2\log(2)\, I(Z;W)}. \qquad (6.11) \end{aligned}$$

Furthermore, we have,

$$\begin{aligned} & I(\boldsymbol{S};W) \\ =\ & I(Z_1,\cdots,Z_n;W) \qquad (6.12) \\ =\ & H(Z_1,\cdots,Z_n) - H(Z_1,\cdots,Z_n|W) \qquad (6.13) \\ =\ & \sum_{i=1}^{n} H(Z_i) - \sum_{i=1}^{n} H\left(Z_i|\boldsymbol{S}^{i-1}, W\right) \qquad (6.14) \\ \geq\ & \sum_{i=1}^{n} H(Z_i) - \sum_{i=1}^{n} H(Z_i|W) \qquad (6.15) \\ =\ & \sum_{i=1}^{n} \left[H(Z_i) - H(Z_i|W)\right] \qquad (6.16) \\ =\ & \sum_{i=1}^{n} I(Z_i;W) = nI(Z;W), \qquad (6.17) \end{aligned}$$

where $\boldsymbol{S}^{i-1} = [Z_1,\cdots,Z_{i-1}]$. (6.14) follows from the independence of the $S_i's$ and (6.15) holds true since conditioning reduces the entropy. Therefore, we have:

$$I\left(Z;W\right) \leq \frac{I\left(\boldsymbol{S};W\right)}{n}. \tag{6.18}$$

On the other hand, we use recursively the SDPI inequality for the Markov chain in Fig. 3.1 to obtain:

$$\begin{aligned} I\left(\boldsymbol{S};W\right) = I\left(\boldsymbol{S};\boldsymbol{T}_L\right) &\leq \eta_L I\left(\boldsymbol{S};\boldsymbol{T}_{L-1}\right) && (6.19)\\ &\leq \eta_L\eta_{L-1}I\left(\boldsymbol{S};\boldsymbol{T}_{L-2}\right) && (6.20)\\ &\leq \cdots \leq \left(\prod_{k=1}^{L}\eta_k\right) I\left(\boldsymbol{S};\boldsymbol{S}\right) = \left(\prod_{k=1}^{L}\eta_k\right) H\left(\boldsymbol{S}\right). && (6.21)\end{aligned}$$

Therefore, we have:

$$\begin{aligned} &\left|GEN\left(P_Z,P_{W|\boldsymbol{S}}\right)\right| \\ \leq\ &\sqrt{\left(\prod_{k=1}^{L}\eta_k\right)\frac{2\log\left(2\right)H\left(\boldsymbol{S}\right)}{n}} && (6.22)\\ \leq\ &M^{\frac{L}{2}}\sqrt{\frac{2\log\left(2\right)H\left(\boldsymbol{S}\right)}{n}} && (6.23)\\ \leq\ &\exp\left(-\frac{L}{2}\log\frac{1}{M}\right)\sqrt{\frac{2\log\left(2\right)H\left(\boldsymbol{S}\right)}{n}}, && (6.24)\end{aligned}$$

where $0 \leq M = \max_{k=1,\cdots,L}\eta_k \leq 1$. This completes the proof of Theorem 2.