2021

Artificial Intelligence Aided Receiver Design for Wireless Communication Systems

Wenjie Xu
xu51@live.marshall.edu

ARTIFICIAL INTELLIGENCE AIDED RECEIVER DESIGN FOR WIRELESS COMMUNICATION SYSTEMS

A thesis submitted to
the Graduate College of
Marshall University
In partial fulfillment of
the requirements for the degree of
Master of Science in Engineering
In
Electrical and Computer Engineering
by
Wenjie Xu
Approved by
Dr. Wook-Sung Yoo, Committee Chairperson
Dr. Imtiaz Ahmed
Dr. Pingping Zhu
Dr. Ramesh Annavajjala

Marshall University
August 2021

APPROVAL OF THESIS

We, the faculty supervising the work of Wenjie Xu, affirm that the thesis, *Artificial Intelligence Aided Receiver Design for Wireless Communication Systems*, meets the high academic standards for original scholarship and creative work established by the M.S.E. in Electrical and Computer Engineering and the Department of Computer Sciences and Electrical Engineering. This work also conforms to the editorial standards of our discipline and the Graduate College of Marshall University. With our signatures, we approve the manuscript for publication.

Yoo, Wookh

Dr. Wook-Sung Yoo, Department of Computer Science and Electrical Engineering — Committee Chairperson — Date 07/12/2021

Imtiaz Ahmed

Dr. Imtiaz Ahmed, Department of Electrical Engineering and Computer Science, Howard University, Washington, DC — Committee External Member — Date 07/12/2021

Pingping Zhu

Dr. Pingping Zhu, Department of Computer Science and Electrical Engineering — Committee Member — Date 07/12/2021

A. Ramesh

Dr. Ramesh Annavajjala, Engineering Department, College of Science and Mathematics, University of Massachusetts Boston, Boston, MA — Committee External Member — Date 07/12/2021

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my supervisor, Dr. Imtiaz Ahmed, for his patient, constant guidance, insightful comments, and immense knowledge, and for providing invaluable support all the time throughout this thesis. I am extremely lucky to have an opportunity to work with him. Not only has he encouraged me with numerous help for my research, his friendship, insight, and wisdom deeply inspired me and motivated me to adjust my way of thinking. I could not have accomplished a certain goal without him.

My sincere thank goes to my other supervisor, Dr. Wook-Sung Yoo, for his consideration, motivation, continuous guidance, an indispensable support. Every piece of advice that he gave me indicated the correct direction of the navigation of my thesis. He continuously taught me how to express my opinion logistically. I am extremely fortunate to obtain his advising throughout this thesis.

I am highly grateful to my thesis committee, Dr. Pingping Zhu, Dr. Ramesh Annavajjala, for their constructive feedback and advice. Their perceptive comments helped me to further revise and rectify the insufficient of my thesis. Their plentiful academic experiences and insights provided versatile potential direction for my research.

Last but not the least, I would like to thank my father Dr. Shuye Xu, my mother Prof. Yujuan Wei, my boyfriend PingHsuan Chan, and all of my friends. During all these years, they provided incredibly loving, patient, encouragement, and support to me. I have been enabled to attempt various challenges with their suggestions and motivation. Without their love and accompany, I would have not been who I am.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Physical layer (PHY) design in the wireless communication field realizes gratifying achievements in the past few decades, especially in the emerging cellular communication systems starting from the first generation to the fifth generation (5G). With the gradual increase in technical requirements of large data processing and end-to-end system optimization, introducing artificial intelligence (AI) in PHY design has cautiously become a trend. A deep neural network (DNN), one of the population techniques of AI, enables the utilization of its 'learnable' feature to handle big data and establish a global system model. In this thesis, we exploited this characteristic of DNN as powerful assistance to implement two receiver designs in two different use-cases. We considered a DNN-based joint baseband demodulator and channel decoder (DeModCoder), and a DNN-based joint equalizer, baseband demodulator, and channel decoder (DeTecModCoder) in two single operational blocks, respectively. The multi-label classification (MLC) scheme was equipped to the output of conducted DNN model and hence yielded lower computational complexity than the multiple output classification (MOC) manner. The functional DNN model can be trained offline over a wide range of SNR values under different types of noises, channel fading, etc., and deployed in the real-time application; therefore, the demands of estimation of noise variance and statistical information of underlying noise can be avoided. The simulation performances indicated that compared to the corresponding conventional receiver signal processing schemes, the proposed AI-aided receiver designs have achieved the same bit error rate (BER) with around 3 dB lower SNR.

Keywords: Deep Learning; Receiver Design; Multi-label Classification; Bit Error Rate

CHAPTER 1

INTRODUCTION

Background

Since the first-generation network has been launched in 1983, wireless communication technology has improved dramatically and entered fifth-generation (5G) rapidly in recent years (Ly & Yao, 2021). Numerous applications of 5G have emerged in different layers, for instance, massive Internet of Things (IoT), enhanced mobile broadband communication system (eMBB), ultra-reliable and low latency communication (URLLC) in the communication network field, as well as healthcare tech, assisted wearable device, financial technology, and smart home and transportation systems which can be effortlessly seen in our daily life in last few years (Agiwal et al., 2016). 5G is an enlightened application of leading-edge wireless communication on the physical layer (PHY) design with a few representative characteristics, such as requiring innumerable data, high speed and data rate, and low latency.

Artificial intelligence (AI) tool gradually penetrates all walks of life in recent years and undoubtedly a widespread technique not only applied in the Computer Science field, for instance, visualizations, speech processing, image recognition, etc., but also rapidly utilized in wireless communication realm (Aldossari & Chen, 2019; O'Shea et al., 2017; Qin et al., 2019). However, sophisticated techniques are increasingly required numerous datasets which are difficult to handle manually, e.g., 5G, massive multiple-input multiple-output (MIMO) (Huang et al., 2019), etc. To deal with this problem, a common trend is to apply AI tools to cooperate with conventional algorithms in wireless communication systems. For instance, compared to the conventional approach of signal detection and classification, a well-trained DNN requires a much shorter processing time (e.g., several milliseconds) to complete the same tasks (National

Instruments, 2019). Machine learning (ML), a well-known application of AI technology, allows machines to act correspondingly through learning from numerous data without explicit programming (C. X. Wang et al., 2020). One of the notable implementations of ML is deep learning (DL) or deep neural network (DNN), which applies multiple layers in neural networks to classify patterns through training sample data (Marcus, 2018).

Although existing communication algorithms have plenty of mature and optimal applications on communication system design, fulfilling incremental complex PHY layer designs with AI-assisted communication structure, like the 5G system, has the following reasons.

- In conventional communication system structure, there are various blocks, such as channel coding and decoding, modulation and demodulation, equalization, etc., which are responsible for processing transmitted or received signals at different stages. In particular, these blocks usually handle their tasks individually without intervention on the functions of other blocks. This block structure has a significant capacity of optimizing performance in the current signal processing block to solve various imperfections producing from practical channels; however, an optimal algorithm for global system processing is still a challenge for researchers (Wang et al., 2017). Thus, unlike the mode of sub-optimal realization for conventional block by the block communication system, DL attracts more attention since its ability to break through the limitation of block-based structure to optimize the property of an end-to-end system (O'Shea & Hoydis, 2017; Wang et al., 2017). Furthermore, another remarkable feature of DNN is the ability to process a large amount of data, and hence it is robust to cope with interferences from the original blocks and channels.

- Modeling wireless communication channels mathematically with harsh conditions are hard to implement. In conventional communication, designing a system model relies on accurate mathematical modeling of each block (Qin et al., 2019). However, in practical scenarios, potential unknown impact in developing complex models that result in analytical system representation becoming strenuous (Qin et al., 2019). For example, a complicated channel that cannot be handled by Maxwell's equation is hard to express analytically through a rigid framework, such as molecular or underwater communications (Farsad & Goldsmith, 2017; Wang et al., 2017). Moreover, extra nonlinearities and imperfections, which demand more robust signal processing algorithms to realize low consumption of systems, are introduced to the current system, and yet it could cause high computational complexity (O'Shea & Hoydis, 2017; Wang et al., 2017). In this case, DNN is expected since its learning algorithm can achieve the optimization of end-to-end performance without the requirement of an exact system model mathematically (Wang et al., 2017). In addition, based on concurrent structures of DNNs, low accuracy data can be utilized in executing DNN with low energy consumptions and high computational speed (O'Shea & Hoydis, 2017).

The open systems interconnection (OSI) model is a theoretical framework and was developed to intuitively describe a functional network. Seven fundamental layers in the OSI model altogether exhibit the process of transmitting data globally (GeeksforGeeks, 2020b). In the wireless communication field, unlike the initial application of ML on higher layers of the OSI model, such as managing resources, researchers pay significant attention to DL implementation on PHYs (the first layer of the OSI model) (Sattiraju et al., 2019). Generally, the block-based design approach is fundamental at transmitter and receiver, and this approach displays an optimal

performance. In addition, a MIMO system can provide a wider link range and higher data throughput without additional transmitted power and bandwidth and hence is widely utilized in the communication realm (Kashyap & Bagga, 2014). However, due to the generating interferences and fading between each block and every channel for multiple received ends, the optimal performance of an end-to-end system might not be guaranteed by a block-based design approach. Moreover, whereas some existing algorithms of signal processing offer optimal and robust performance, they often entail high computational complexity. All these points motivate researchers in the wireless communication field to push the boundaries of throughput and bit error rate performance for the end-to-end system. Note that our purpose of deploying AI tools on physical layers is not to reinvent the conventional signal processing algorithms but to facilitate the supplement and perfection of existing design.

Contributions

The primary contributions of this thesis are enlisted as follows:

- Developing two fully-connected DNN-based functional signal processing blocks with different diversity receiver cases.
- Training these DNN models offline with various datasets that comprise different SNR values, different sorts of noises, channel fading, etc., and deploying the trained models in real-time applications.
- Demonstrating the effectiveness of the DNN based-functional blocks by comparing BER performances to corresponding conventional signal processing schemes.

Organization of The Thesis

The remaining work is organized as follows:

In Chapter 2, the theoretical background of the conducted research is introduced. In this chapter, the process of channel coding techniques and baseband modulation is presented. Furthermore, equalization is illustrated for multiple antennas cases. In addition, an overview of DNN and multi-label classification (MLC) algorithms are given.

In Chapter 3, we propose a fully connected DNN model with an MLC manner as a joint demodulator and decoder (DeModCoder). We train our DeModCoder offline through a wide range of SNR and apply online to investigate bit-error-rate (BER) performance. We arrange conventional demodulation collaborating with syndrome-based decoding algorithm, and multi-output classification (MOC) manner as two baseline schemes to evaluate the performance of MLC DeModCoder.

In Chapter 4, based on the implemented MLC-based DeModCoder and following the same experimental structure, we develop a joint equalizer, demodulator, and channel decoder (DeTecModCoder) utilizing DNN with MLC algorithm. In this work, we consider a D-diversity receiver system. Moreover, in addition to additive white Gaussian noise (AWGN), we introduce non-Gaussian noise and channel fading to the transmitted signal in order to enhance the robustness of the system through adding noise and interference.

Finally, the conclusion of this thesis and future work are presented in Chapter 5.

CHAPTER 2

THEORETICAL BACKGROUND

The theoretical background is introduced in this section, including how the channel decoding and baseband demodulation work on the receiver side, and what type of deep neural network is utilized in our design. The block diagram of the digital communication system is shown in Figure 1.

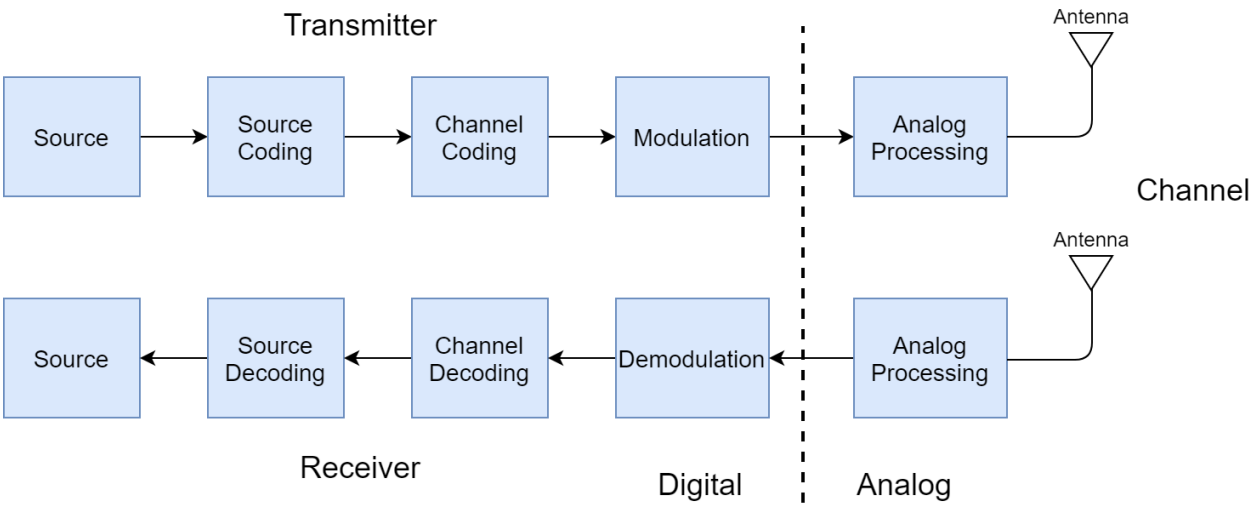

Figure 1. Block Diagram of Digital Communication System.

Channel Coding Process

In 1948, the channel coding technique was initiated by Claude Shannon (Costello & Forney, 2007). This technique is an essential process in wireless communication systems and is also represented as a forward error control coding technique. Channel coding is applied in the digital communication stage and utilized to detect erroneous bits in order to correct them before modulation. Not only at the transmitter end but also at the receiver end arrange this process, called channel decoding. Since the potential of transmitted data bits is corrupted by various noise, interference, and fading, raw information bits are supposed to be protected and thus parity

bits, as known as redundant bits, are added to the original data bits (Faruque, 2016). Therefore, at the receiver end, the corresponding channel decoder can detect and correct error bits through the transmitted parity bits. However, extra redundancy bits cost a higher price in bandwidth (Grami, 2016). Although we can add adequate parity bits to obtain lower enough BER, the whole bitstream requires more bandwidth (Faruque, 2016). Due to equitable resource allocation, it is crucial to balance the requirement of bandwidth and thus transmission rate is maintained under channel capacity (Chan, 1997).

In order to approximately imitate a practical environment, both Gaussian and non-Gaussian noise are considered to obtain noisy transmitted signals. Received signals, which spread through multiple channels, are impaired by channel fading at the received end. Within this process, noise and fading are two basic influences that occur in wireless communication. Different from the point-to-point wired communication system, the transmission process happens through the air with a variety of interferences for wireless communication since distortions can exist from single or multiple transmitters to a common receiver or multiple receivers, e.g., uplink or downlink of a cellular system respectively; whereas fading is the fluctuation of channel strength over time because of path loss, obstacles influences of weather or shadowing, and multipath propagation (Tse & Pramod, 2005). In particular, Rayleigh fading is a typical non-light-of-sight component that occurred far range between transmitter and receiver (Clerckx & Oestges, 2013). Gaussian noise and non-Gaussian noise are regarded as mathematical noise models based on the distributions they are following, e.g., Gaussian noise follows Gaussian distribution. In Chapter 4, we consider two types of non-Gaussian noises containing generalized Gaussian noise (GGN) and ϵ-mixture noise (Ahmed, 2014).

Hamming Code and Reed-Solomon Code

Our experiment is to design a DNN-based MLC joint baseband processing model to improve BER performance and reduce computational complexity compared with syndrome decoding algorithms (Grami, 2016). In this research, Hamming Code and Reed-Solomon Code are utilized to demonstrate simulation results of conducted design.

Hamming Code is published by Richard Hamming in 1950 (Costello & Forney, 2007). (n, k) Hamming code is one of the linear block codes with n length of bits and k information bits, thus $n - k$ parity bits are mapping into the code words (Woods, 2012). With a similar concept of parity check method, Hamming code applies parity matrix to calculate the location of occurring errors and correct them(Garg, 2007). For linear block code, only '1' or '0' might occur in each bit position. Therefore, an (n, k) linear block code contains 2^n possible received 7-bit code stream, where 2^k of them are valid code bits (Parker, 2017). For instance, an (7,4) Hamming code exists 2^7 possibilities of received bit sequences, and 2^4 of them are regarded as valid code bits. On linear block codes, an essential concept of error-correcting capability is Hamming distance, which is regarded as the number of positions for the differences between two respective codewords (Djordjevic, 2012). The minimum distance d_{min} for Hamming code is 3 (the Hamming distance is 3), which represents there exists a minimum of 3 different bits between transmitted code bits and other possible code words (Parker, 2017). For instance, on (7,4) Hamming code, every codeword has at least 3 different bits out of the other $2^4 - 1$ codewords, and it is used to correct only a single error.

Reed-Solomon (RS) code is a linear and non-binary block code and is commonly used for its capacity of correcting large errors (Garg, 2007). Unlike adding parity bits for Hamming code, RS code is determined by multi-bit symbols, which are generated by various raw bits (Mitchell,

2009). Each symbol is made up of m bits. Generally, an (n, k) RS code represents n block lengths and k message lengths, where $n = 2^m - 1$, $k = n - 2t$, in which t is the maximum number of error-corrected by each symbol (Klima et al., 2020). Therefore, $2t$ parity bits are added to the (n, k) RS code. The minimum distance of RS code is represented as $d_{min} = n - k + 1$, where k is the total amount of symbols for n lengths of code bits (Djordjevic, 2012). To generally overview the decoding process of RS code, we assume message polynomial $P(x)$, generator polynomial $G(x) = [(x - \alpha)(x - \alpha^2) \cdots (x - \alpha^{2t})]$, where t is the capacity of correcting error and α is a primitive root of Galois Fields (GF), which consists of finite elements (Bhaskar, 2020; Westall, 2010). If the quotient of the received signal $R(x)$ divided by $G(x)$ is equal to 0, there is no error emerging; otherwise, the received signal $R(x)$ can be formed as $R(x) = P(x) \cdot G(x) + e(x)$, where $e(x)$ expresses error polynomial containing the information of error positions and the number of error (Bhaskar, 2020). It is worth mentioning that within the decoding stage, RS code enables to correct of multiple symbol errors. Based on symbol-based structure, the decoder corrects the complete symbols rather than a single error bit (Mitchell, 2009). Therefore, for burst error correction causing by fading, (n, k) RS code shows a robust performance than binary code although it provides high computational complexity (Mitchell, 2009).

Baseband Modulation

Normally, raw information signals with a low frequency that constitute frequencies from near 0 Hz up to some specific low value, is called baseband signal (Sundareshan, 1992). Since directly transmitting multiple signals with a similar spectrum of frequency could cause interference, they are required to be modulated before passing channels. By modulation process, baseband signals are transferred to carrier waves with high frequency (Plonus, 2001). Phase Shift

Keying (PSK) modulators are powerful and popular digital modulators; thus, binary phase-shift keying (BPSK) and quadrature phase-shift keying (QPSK) collaborate with optimal equalizer and syndrome-based channel decoder blocks to constitute conventional scheme in this research.

BPSK and 4-QAM

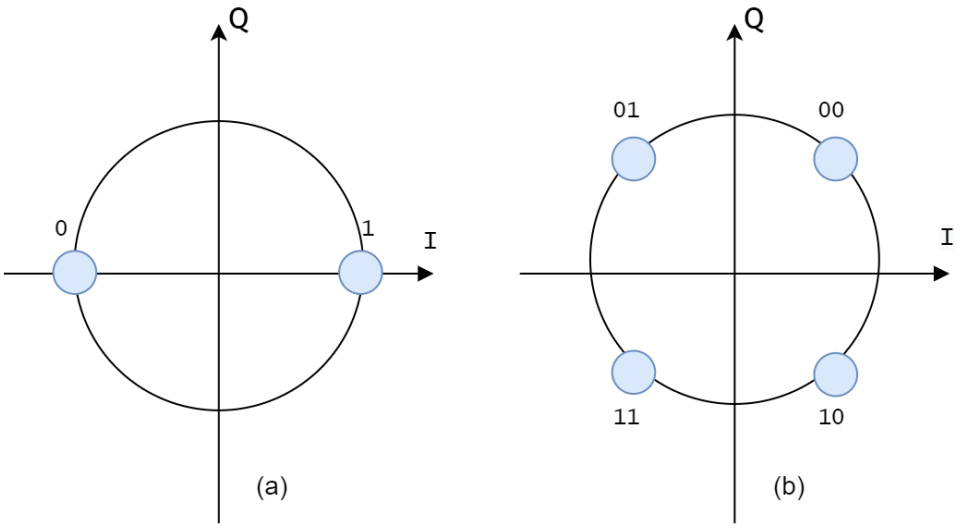

Figure 2. Constellation Diagram of (a) BPSK and (b) 4-QAM.

BPSK is the fundamental instance of PSK, which is one of the most widely applied bandpass modulators (Nassar, 2001). First let us assume the input signal has a form $s(t) = \lambda \cos(\omega t + \theta)$, where phase θ will affect message bits $s(t)$. As can be seen in Figure 2 (a), In-phase (I) axis and Quadrature (Q) consist of a plane, called a constellation, and binary symbols '1' and '0' are represented by phase-shifting of 0 to π with the same amplitude (Gallion, 2016). It is remarkable that the phase of signal only shifts when the binary symbol is changed, e.g., if binary bit changes from '0' to '1', the phase of carrier signal shifts 180 degrees, and thus the signal sensitivity for nonlinearity is drastically reduced (Gallion, 2016). For example, assuming bit sequence '001' goes into BPSK modulator bit-by-bit, output waveform maintains $\lambda \cos(\omega t + 0^o)$ when there is no phase change between first and second bit '0'; in contrast, the waveform adjusts to $\lambda \cos(\omega t + 180^o)$ when the third bit '1' enters (Nassar, 2001). BPSK modulation is

robust and produces good BER performances even with a low range of SNRs; therefore, it is capable of showing good performance in harsh environments (Lopez-Gordo & Pelayo, 2013). However, since the limitation of BPSK is that only 1 bit can be transmitted for each symbol, it may not be appropriate for high data rate requirements (Gallion, 2016).

In Figure 2 (b), by adding two binary symbols to BPSK, we can obtain QPSK, as known as 4-QAM, which can carry double BPSK data rate with less sensitivity to noise (Haque et al., 2012). In particular, PSK and QAM are different types of modulators. However, QPSK and 4-QAM have the same points on the constellation diagram, thus they are identified as the same modulation scheme. Since 4-QAM simultaneously carries 2 bits over every time interval, its spectrum efficiency is two times higher than BPSK (Gallion, 2016). Rather than 0 to π phase shifting of BPSK, phase θ of 4-QAM is switched to 0^o, 90^o, 180^o, and 270^o for bit pair '00', '01', '10', and '11', respectively (Nassar, 2001).

Equalization for Multi-Antenna Cases

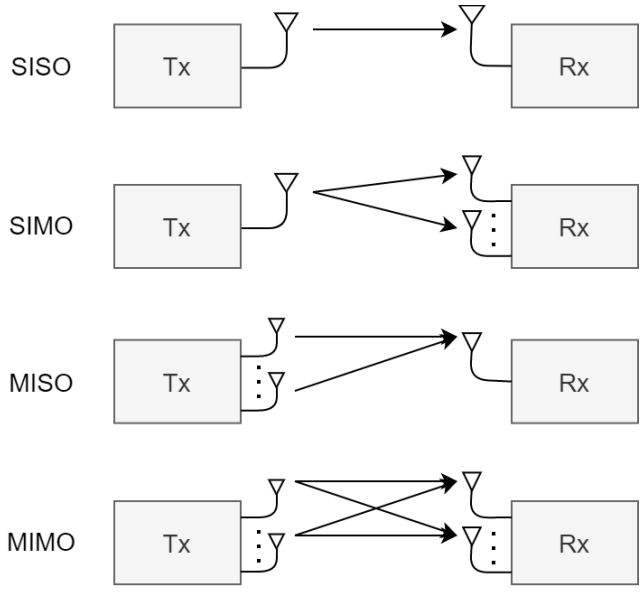

Figure 3. System Structures of SISO, SIMO, MISO, and MIMO.

Figure 3 exhibits four common wireless communication systems, containing single-input single-output (SISO), single-input multiple-output (SIMO), multiple-input single-output (MISO), and multiple-input multiple-output (MIMO). These systems are widely applied in different conditions of the environment. The applications of multiple antenna arrangements significantly break through the limitation of system coverage area and data throughput without additional bandwidth and power consumption (Kashyap & Bagga, 2014). However, intersymbol interference could be introduced to the signal at the receiver due to multipath propagation between different channels. Intersymbol interference, a consequence of delay spread, leads to being indistinguishable between transmitted signal and noise. Not only the effect of noise and intersymbol interference but amplifiers and mixers will further increase the possibility of bit error occurrence by generating nonlinear distortion (Xu et al., 2018). Based on this situation, various types of the equalizer, which is used to add or subtract gain over a selectable frequency scale (Long, 2014), are deployed to diminish the aforementioned nonlinearity and influences. On the other hand, accurate channel state information (CSI) is essential for massive MIMO system design to achieve desirable performance. However, it is difficult to execute CSI estimation and feedback in such a system due to the complexity of obtaining a precise channel model (H. He et al., 2019). In this case, these findings motivate researchers to apply machine learning techniques to deal with this problem.

Deep Neural Network and Multi-Label Classification

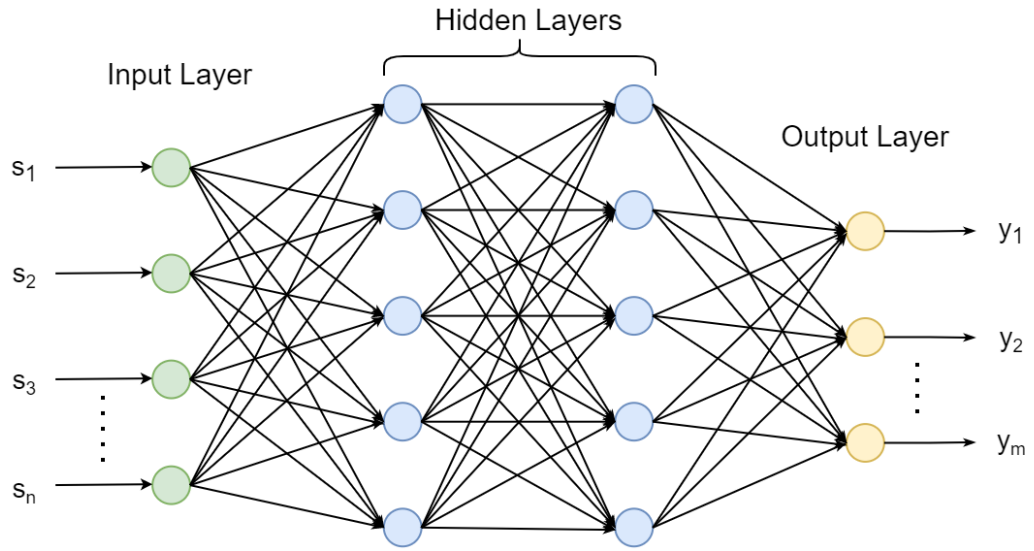

Figure 4. Structure of Deep Neural Network.

DNN has a powerful capability on the task of visualization processing (Mahmood et al., 2017), and researchers also extend this powerful tool into the wireless communication field. In general, except input layer and output layer, DNN contains multiple hidden layers with parameters of weight W, bias b, and activation function $g(x)$ (Witten et al., 2017). As can be seen in Figure 4, the structure of DNN imitates the human brain consisting of massive nodes. We assume that the total number of input and output nodes are N and M respectively. Input and output data formed as a column matrix s_n and y_m, where $n \in \{1,2,\cdots,N\}$, $m \in \{1,2,\cdots,M\}$, respectively. Without any calculation, the input layer passes data symbols to hidden layers, and the output layer receives data bits from hidden layers. At each neuron, transmitted data or signals are processed through an activation function, which introduces nonlinearities to the output of current neurons through sum the weight and adds a bias in order to have the ability to handle complicated tasks by improving the learning capability of NNs (GeeksforGeeks, 2020a). In this research, rectified linear unit (ReLU) and parametric rectified linear unit (PReLU) activation functions are utilized in our design. ReLU, with function $g(x) = \max(x, 0)$, gives the output

which is the same as input when it is a positive value, otherwise, ReLU gives 0 as the output of neurons. In addition, ReLU is set as a default activation function of most of NNs since it often attains good performance with low computational complexity and short training duration (Brownlee, 2019). However, in some cases, ReLU outputs 0 when given inputs maintain negative due to updating large weights (Brownlee, 2019). Therefore, we considered PReLU to further increase the performance of our model avoiding the limitation of the ReLU activation function. PReLU can be formed as $g(x) = \max(x, \alpha x)$, where $\alpha \in (0,1)$. Instead of output 0 for negative value, PReLU introduces a coefficient α to adjust the slope of negative input values so that 'dying ReLU' occurring in ReLU can be avoided (K. He et al., 2015). Furthermore, we applied a simple fully connected neural network with a modified number of hidden layers and neurons to demonstrate the simulation results compared with our designed baseline schemes, which are introduced in detail in Chapter 3 and Chapter 4.

A fully connected neural network consists of a bunch of layers that are fully connected. Every output of the current neuron relies on every input dimension from the previous layer. The reason for selecting a fully connected neural network is that there is no need to assume a particular input based on the "structure agnostic" characteristic of the fully connected neural network, e.g., the input dataset contains special noise, distortion, etc. (Ramsundar & Zadeh, 2018).

Whereas a common concept of conventional design is to mathematically model a system block-by-block, AI-based methods enable to learn from input datasets (S), which should be mapped to particular output datasets (Y), and thus design functions of a network (Sattiraju et al., 2019). If both input datasets S and output datasets Y are required by the model to satisfy the function $Y = f(S)$, the manner of this model is called supervised learning (Brownlee, 2020b). In

our user cases, we apply a supervised learning approach to train the DNN. Thus, information bit sequences, which are generated from a conventional channel processing scheme, are identical to output datasets to train DNNs. Once trained, the output of a trained DNN can be used to compare with the information datasets in order to evaluate the BER performance. Therefore, we enable to predict output datasets from different input variables.

In the simulation process, we applied a backpropagation algorithm to improve the performance of NN (M. L. Zhang & Zhou, 2006). An essential question is how we handle the non-continuous ReLU when we are using the backpropagation algorithm. In most of the cases, ReLU has a gradient, where output 1 for $x > 0$ and output 0 for $x < 0$ (Brownlee, 2019). When $x = 0$, the ReLU function is non-continuous and non-differentiable; however, it does not mean that ReLU cannot be used in practice. First, when the neural network (NN) is trained, it does not usually occur that a local minimum value is reached for cost function, and the non-differentiable points are seldom; therefore, the undefined gradient is limited in a small threshold and be acceptable (Sarkar, 2018). Second, in the practical application of TensorFlow, instead of warning an error of undefined deviation, it usually returns one of the one-sided deviations (Sarkar, 2018). Hence, the software usually outputs 0 as the deviation of ReLU for $x = 0$ or the non-linear part (Brownlee, 2019), and this will not affect the calculation of the backpropagation algorithm.

Multi-Label Classification

After training through hidden layers, we introduce the multi-label classification (MLC) method in the output layer where each label comprises of binary class using Keras from the TensorFlow library (Abadi et al., 2016). Based on this arrangement, transmitted data bits are presented obviously and used to calculate BER directly. A typical classification problem has one or more than one mutually exclusive class, where a set of inputs corresponds one each one of

them (Brownlee, 2020a). Unlike traditional classification problems, MLC generates grouping labels for every object (Luaces et al., 2012). These labels are not mutually exclusive with simultaneous requirements, and thus we can simply use binary bitstream generated from the MLC output layer to compare with original information bits. Therefore, the first-order strategy in the MLC technique is applied, where there is no need to consider coexistence for each label (M. L. Zhang & Zhou, 2014).

CHAPTER 3

JOINT DEMODULATION AND DECODING WITH MULTI-LABEL CLASSIFICATION USING DEEP NEURAL NETWORKS

Background Overview

Artificial intelligence (AI) technologies in the physical layer of wireless communication systems have recently sparked a burst of interest. A range of recent research indicates that AI tools, such as DNNs, can improve or substitute various physical layer algorithms in transmitter and receiver chains (Akın et al., 2020; Qin et al., 2019; Ye et al., 2018). There is a high possibility to gather plenty of datasets, which used to train AI tools, because of diverse application fields of current communication systems, such as fifth-generation (5G) new radio (NR) (Dahlman et al., 2018), 802.11ax wireless LAN (Wi-Fi 6) (Y. Zhang et al., 2020), etc., which cause ultra-high throughput. In general, the conventional transmitter and receiver designs are block-based, with each block performing its task optimally and often efficiently. However, for different communication systems, block-based design manner may not guarantee optimal end-to-end performance in terms of energy efficiency, throughput, error rate, etc. Moreover, notwithstanding that some of the current signal processing algorithms provide robust and optimal performance, they often result in high computational complexity. These findings altogether prompted the wireless communication engineers to adopt AI techniques to push the boundaries of the error rate and throughput performances while limiting the complexity of operation and computation within a particular threshold. It is worth mentioning that the objective of deploying AI tools in the physical layer of wireless communications is not to replace the existing signal processing algorithms without a specific goal but to complement the existing design with more

sophisticated algorithms that may outperform the conventional design and/or alleviate the implementation complexity.

It has recently become apparent that in a variety of use cases, deploying AI in equalization (detection), demodulation, and decoding improves performance significantly compared to conventional solutions. One collection of research works concentrates on assisting the conventional algorithms with AI techniques in order to further improve the throughput or minimize the error rate performance (H. He et al., 2018; Y. He et al., 2020; Nachmani et al., 2016, 2018; Sun et al., 2020). In contrast, some researchers focus on substituting signal processing blocks with AI tools, such as fully connected DNN, recurrent neural network (RNN), convolutional neural network (CNN), autoencoder, etc. (Akın et al., 2020; Farsad & Goldsmith, 2017; Mohammad et al., 2018; Vaz et al., 2019; H. Wang et al., 2019; Ye et al., 2018).

DNN based channel decoder designs for linear block codes have been well investigated in the following research and reference therein. According to Vaz et al. (2019) to decode a (7,4) Hamming code, a fully connected DNN-based decoder was developed. The DNN-decoder was trained offline using the backpropagation algorithm and then used to decode the bitstream online at the receiver to retrieve the information bits. Research work from Nachmani et al. (2016, 2018) enhanced the belief propagation process by assigning weights to the edge of the Tanner graph for decoding block codes. Deep learning (DL) based bit interleaved coded modulation (BICM) receiver is designed by Y. He et al. (2020) for low-density parity-check (LDPC) coded direct current-biased optical orthogonal frequency division multiplexing (DCO-OFDM) systems. The conditional probabilities for the log-likelihood ratio (LLR) detector are first predicted using a non-iterative neural network (NN) assisted BICM receiver. Two iterative NN-aided BICM

schemes were designed to improve LLR efficiency in different flat and frequency selective fading channels.

In this chapter, we design a joint baseband demodulator and channel decoder (DeModCoder) for a communication receiver with a fully connected DNN (Goodfellow et al., 2016) using a multi-label classification (MLC) algorithm while incorporating binary class for each label (Brownlee, 2020a; Grunau et al., n.d.; M. L. Zhang & Zhou, 2006). The designed DL-assisted (DNN based) DeModCoder is applicable for linear digital modulation schemes, e.g., binary-phase shift keying (BPSK), M-ary quadrature amplitude modulator (M-QAM), etc. and linear block codes, e.g., Hamming code, Reed Solomon (RS) code, etc. We aim to investigate and compare the error rate performances of the designed DeModCoder for different channel coding and digital modulation schemes with those obtained from conventional demodulators and decoding schemes. For the design phase, we consider a variety of scenarios that represent a variety of use cases in different wireless communication systems and standards. In addition, we stress that the use of MLC-based DeModCoder lessens the computational complexity compared to the commonly deployed multiple output classifier (MOC) scheme in the literature from Ahmed & Allen (2020). We train a DNN over a wide range of training symbols gathered from various signal-to-noise ratios (SNRs) to further diminish computational complexity and demonstrate the effectiveness of the designed MLC DeModCoder over the conventional approach and MOC DeModCoder by simulations.

The remainder of this chapter is organized as follows. We describe the system model and the DL-assisted DeModCoder in Sections 3.2 and 3.3, respectively. The simulation results are presented in Section 3.4 and the conclusions are made in Section 3.5.

System Model

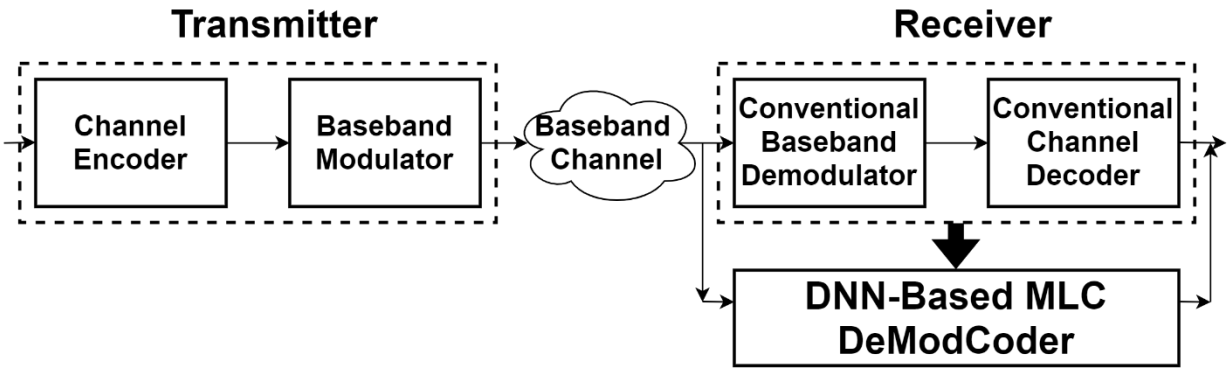

Figure 5. A Point-to-Point Communication Link with Conventional vs. AI Based Receiver Processing.

We consider a point-to-point communication system (shown in Figure 5) with a transmitter and a receiver, where the transmitter deploys (n', k') channel coding and baseband digital modulation before sending the transmit signal to the receiver. We assume that the transmitter and the receiver are perfectly synchronized[1]. Let us denote channel encoding specific parameters $n = \mathcal{T}n'$ and $k = \mathcal{T}k'$, where $\mathcal{T}$, being an integer, represents the number of bits used to represent an information symbol for a given channel coding scheme[2]. For instance, we set $\mathcal{T} = 1$ and $\mathcal{T} = \log_2(n' + 1)$ for (binary) Hamming code and RS code, respectively. The transmitter sends a message d containing k information bits (e.g., $k = k'$ for (binary) Hamming code and $k = \log_2(n' + 1)\, k'$ for RS code) to the receiver over a noisy channel. At first, k information bits are encoded with (n', k') linear block code and represented as $x = f(d)$ with n bits (e.g., $n = n'$ for (binary) Hamming code and $n = \log_2(n' + 1)\, n'$ for RS code). Here, $f(\cdot)$ denotes the channel encoding techniques for linear block codes. The encoded bits are then passed

[1] In particular, we assume that the timing and frequency errors are estimated and corrected with appropriate signal processing techniques. Furthermore, we assume that the channels are estimated and equalized as well.
[2] In particular, $\mathcal{T}'$-ary ($\mathcal{T} = \log_2 \mathcal{T}'$) information symbols (for the purpose of channel coding) can be constructed from Galois Field (GF) $2^{\mathcal{T}}$ (Proakis & Salehi, 2001)

through a baseband digital modulator, e.g., BPSK, M-QAM, etc., and hence the modulated symbols are denoted as $s = g(x)$, where $g(\cdot)$ represents the baseband modulation process. It is worth pointing out that s is the baseband transmitted signal taken from a finite M-ary modulation symbol alphabet, c.f., 4-QAM, 16-QAM, BPSK, etc. with total power $\mathcal{E}\{|s|^2\} = P_s$. Note that $\mathcal{E}\{\cdot\}$ denotes statistical expectation operation. The transmitter communicates with the receiver over I equal-duration transmission time intervals. The discrete-time baseband modulation model for a given time interval i can be expressed as

$$y[i] = s[i] + w[i], i \in \{1,2,\cdots,I\} \quad (1)$$

where $y[i]$ is the (complex) received signal and $w[i]$ represents complex-valued additive white Gaussian noise (AWGN) with zero mean and variance σ^2. The received SNR is defined as $\gamma = P_s/\sigma^2$. Note that although we consider an AWGN baseband channel model in this chapter, our developed MLC DeModCoder is applicable for any fading channel, where the signals at the receiver need to be equalized first before feeding them to the DeModCoder block.

The received signals are grouped in blocks and fed into the DNN based MLC DeModCoder, which outputs the decoded message bits. It is worth noting that the designed DeModCoder jointly demodulates and decodes the received signal in a single module (see the receiver block in Figure 5), which would otherwise be demodulated and decoded in two consecutive (signal processing) blocks. The MLC DeModCoder is trained offline using a few input datasets containing $y[i]$ and corresponding output datasets $d[i]$ that encompass a wide range of SNRs. We discuss the details of the procedure in Section 3.3.

DL Assisted Joint Demodulation and Decoding

In this section, we demonstrate how to develop a DL-assisted joint demodulation and decoding scheme. As mentioned in Section 3.2 we train a DNN with received (noisy) complex baseband signal as its input and transmitted information data bits as its output.

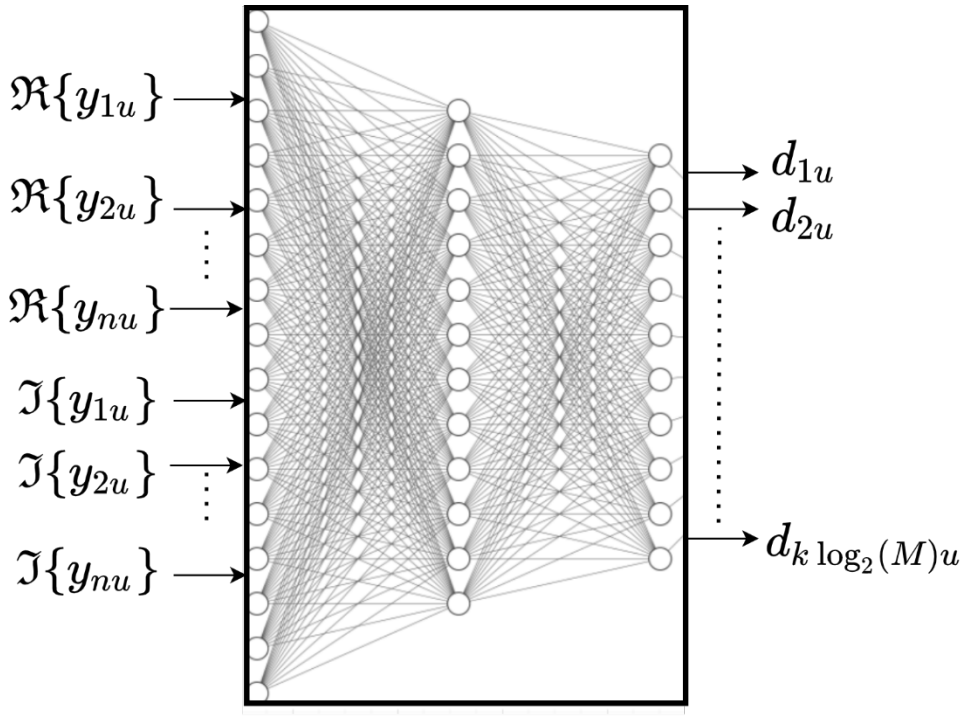

Figure 6. DNN based MLC DeModCoder.
The size of input and output training symbols for a given epoch $\boldsymbol{u}$ are $\boldsymbol{2n}$ and $\boldsymbol{k\log_2(M)}$, respectively for $(\boldsymbol{n'}, \boldsymbol{k'})$ linear block code and $\boldsymbol{M}$-ary linear baseband modulation scheme. Note that $\boldsymbol{n = n'}$ $(\boldsymbol{n = \log_2(n'+1)\, n'})$ and $\boldsymbol{k = k'}$ $(\boldsymbol{k = \log_2(k'+1)\, k'})$ for Hamming code (RS code). $\mathfrak{R}\{\cdot\}$ and $\mathfrak{I}\{\cdot\}$ denote real and imaginary parts of a complex variable.

Design of DNN Based MLC DeModCoder

As shown in Figure 6, the DNN is equipped with multiple labels at the output, and we apply the MLC scheme to train the DNN with multiple labels at the output in a supervised learning approach (M. L. Zhang & Zhou, 2006). By clearly defining the number of targeted labels as the number of nodes in the output layer, MLC can be leveraged directly from the inherent structure of DNN (Brownlee, 2020a). Generally, classification problems involve

predicting a number of classes using a single label. On the other hand, the output labels in the MLC system are not mutually exclusive (i.e., all output labels are functions of all inputs) and thus collectively represent a variety of classes. A common trend in designing the MLC problem is to design the output layer of DNN as an array of binary indicators. As a result, the binary combination of the array at the output of DNN indicates one of the classes of the given problem.

Construction of Training Datasets: The number of epochs for the training is assumed to be U, where each epoch containing V number of batches. Note that $V = 1$ reflects all the training symbols to be used during a given epoch. Let us denote a set of message data bits as $\mathcal{D}_u$ in a given epoch $u \in \{1, 2, \cdots, U\}$ that the transmitter generates for the purpose of training DNN based MLC DeModCoder. These data bits are mapped to (channel) coded data bit sequence $\mathcal{X}_u$ and then to (baseband) modulated data symbols $\mathcal{S}_u$, $u \in \{1, 2, \cdots, U\}$. Once modulated, the data symbols[3] are then corrupted with complex-valued AWGN samples following (1) and hence we obtain the received sequences $\mathcal{Y}_u$, $u \in \{1, 2, \cdots, U\}$. We set $\{\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_U\}$ and $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_U\}$ as training input and output, respectively.

In particular, $\mathcal{Y}_u$, $u \in \{1, 2, \cdots, U\}$ is represented by $[\Re\{y_{1u}\}, \Re\{y_{2u}\}, \cdots, \Re\{y_{nu}\}, \Im\{y_{1u}\}, \Im\{y_{2u}\}, \cdots, \Im\{y_{nu}\}]^T$, where $\Re\{x\}$ and $\Im\{x\}$ denote real and imaginary parts of complex variable x. Similarly, $\mathcal{D}_u$, $u \in \{1, 2, \cdots, U\}$ is represented by $\left[d_{1u}, d_{2u}, \cdots, d_{k \log_2(M)u}\right]^T$. Here, y_{zu} and d_{zu} are the samples of the received symbol and transmit information bit, respectively, where $z \in \{1,2, \cdots, n\}$ and $u \in \{1, 2, \cdots, U\}$. It is worth mentioning that $\mathcal{D}_u$ and $\mathcal{Y}_u$ contain $k \log_2 M$ data bits and $2n$ data samples, respectively. Note that the complex-valued y_{zu} is decomposed into real and imaginary parts and hence, the size of input training symbols $\mathcal{Y}_u$, $u \in \{1, 2, \cdots, U\}$ is $2n$. For instance, in case of BPSK and 4-QAM

[3] 'Symbols' and 'signals' are used interchangeably in this chapter.

modulation schemes with (7,4) Hamming code, $\mathcal{D}_u$ contains 4 and 8 bits, respectively. Likewise, $\mathcal{Y}_u$ contains 14 symbols each for BPSK and 4-QAM modulation schemes in conjunction with (7,4) Hamming code. In order to model an MLC-based DeModCoder, we develop a fully connected DNN with $2n$ input nodes and $k \log_2 M$ output labels and train it with input and output training sequences.

Training Arrangements: We now define the set of parameters for the MLC DNN block as $\vec{\eta}_h = \{\zeta_{1h}, \zeta_{2h}, \cdots, \zeta_{\mathcal{N}h}, \theta_{1h}, \theta_{2h}, \cdots, \theta_{\mathcal{N}h}\}$ for a given hidden layer $h \in \{1,2, \cdots, \mathcal{H}\}$. Here, $\mathcal{N}$ represents the total number of neurons in hidden layer h and $\mathcal{H}$ denotes the total number of hidden layers. Note that ζ and θ denote the weight and bias factor, respectively. The training datasets $\mathcal{Y}_u$ and $\mathcal{D}_u$ are used to train the MLC DeModCoder while optimizing $\vec{\eta} = \{\vec{\eta}_1, \vec{\eta}_2, \cdots, \vec{\eta}_{\mathcal{H}}\}$ as following:

$$\eta^* = \arg\min \mathcal{G}(\mathcal{D}_u, \widehat{\mathcal{D}}_u) \tag{2}$$

Where $\mathcal{G}$ is a binary cross-entropy loss function (Grunau et al., 2018) and $\widehat{\mathcal{D}}_u$ is the estimation of $\mathcal{D}_u$ that is the output of the considered MLC DeModCoder. We deploy the softmax activation function at the output layer for MLC (Goodfellow et al., 2016), and optimize (2) with the assistance of training data while using the stochastic gradient descent approach and the backpropagation algorithms (Adam optimizer, etc.) (Goodfellow et al., 2016).

Deployment of Interference Model: Note that the trained DNN based MLC DeModCoder is deployed online for joint demodulation and decoding. The received signals $\{y[i], y[i+1], \cdots, y[i+n-1]\}$ are fed into the MLC DeModCoder that provides $\{\hat{d}_1, \hat{d}_2, \cdots, \hat{d}_{k \log_2(M)}\}$ at its output. Here, $\hat{d}_q$ denotes the decoded bit for transmitted data bit d_q, where $q \in \{1,2, \cdots, k \log_2(M)\}$.

Variants of MLC DeModCoder

Based on the structure of the DNN model and its training approach, we consider two different variants of MLC DeModCoder.

1) MLC Variant 1 -- Single NN DeModCoder with MLC: In this case, we train a single DNN over a large dataset and apply MLC. This dataset contains the input and output training samples over a wide range of γ.
2) MLC Variant 2 -- Multiple NN DeModCoder with MLC: In this case, while applying MLC, we train a DNN for each SNR value. Therefore, we create $\mathbb{M}$ DNN and train them individually for $\mathbb{M}$ values of γ before being deployed in real-time data demodulation and decoding.

Baseline Schemes

In order to compare the performance of deployed MLC DeModCoder with conventional algorithms, we consider two baseline schemes as follows:

1) Baseline Scheme 1: In this baseline scheme, we consider the conventional baseband demodulation scheme and syndrome-based channel decoding algorithm (Proakis & Salehi, 2001). Note that we include this algorithm to compare the performance of DNN with the conventional robust signal processing algorithms.
2) Baseline Scheme 2: In this baseline scheme, we use MOC instead of MLC and hence define it as MOC DeModCoder. Note that while designing a DNN based MOC DeModCoder, we use a single label to represent a single information word at its (DNN) output that results an integer number (representing a class) within a range from 0 to $2^k - 1$. Therefore, the MOC DeModCoder contains M labels at the output. In contrast to MLC, DNN based MOC can be implemented with 2^k output nodes, where each node represents

a unique class. Consequently, unlike MLC, the number of output nodes for MOC grows exponentially with k. For instance, in case of a (7,4) Hamming code and BPSK modulation scheme, 16 (binary) output nodes can indicate either the presence or the absence of 16 classes.

Simulation Result

In this section, we evaluate the bit error rate (BER) performances of the proposed DNN-based MLC DeModCoder for joint processing of demodulation and channel decoding and compare them with the baseline schemes. We demonstrate the results for linear baseband modulation schemes, BPSK and 4-QAM, and for linear block channel coding schemes, Hamming code and RS code. In particular, we consider (7,4) Hamming code and (7,3) RS code throughout the simulation results. It is worth mentioning that the conducted study can be extended for any higher-order baseband modulation schemes, e.g., 64-QAM, 256-QAM, 1024-QAM, etc., and advanced linear block codes, e.g., low-density parity-check (LDPC) codes, etc. We consider the following scenarios based on the combinations of baseband modulation and channel coding schemes.

1) Scenario 1: BPSK modulation and (7,4) Hamming code
2) Scenario 2: QPSK modulation + (7,4) Hamming code
3) Scenario 3: BPSK modulation + (7,3) RS code (with 3 bits of information symbols)
4) Scenario 4: QPSK modulation + (7,3) RS code (with 3 bits of information symbols)

We design our simulation platform in Python framework with Tensorflow/Keras modules in order to develop DNN based MLC and MOC DeModCoders (Abadi et al., 2016). For all the considered scenarios, we generate 10^5 realizations of data bits and noise samples for training purposes and another set of 10^5 realizations of random data bits and noise samples for the

purpose of validation. Moreover, for non-NN-based demodulation and decoding, 10^5 independent realizations of random data bits and noise samples are generated for performance evaluation. In addition, we create the training datasets through Monte-Carlo simulations. It is worth mentioning that we can collect practical datasets from the field experiments in a live network or from the controlled environment in a lab setup and thus use the data in training the NN. Recall that for complex-valued training samples, we extract in-phase and quadrature components and include both of them for the input training sequences. For instance, in case of a 7-points complex-valued input training sample (for a given realization), we take both in-phase and quadrature components and hence, consider 14-points real-valued input training samples. We consider the AWGN channel throughout the simulations and obtain BER for a range of γ (from -10 dB to 10 dB).

Number of Hidden Layers and Neurons in DNN

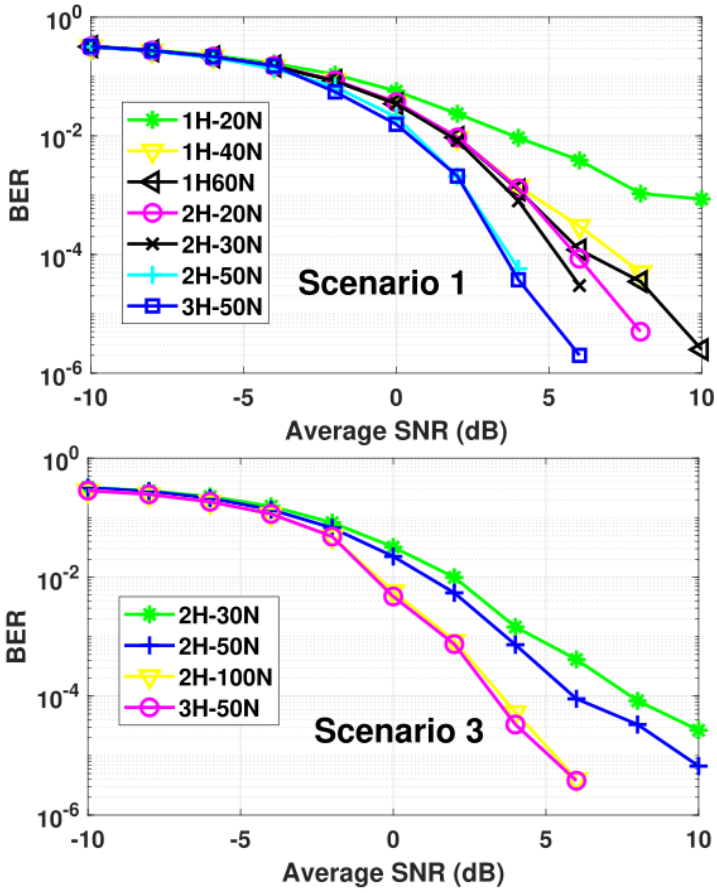

Figure 7. BER vs. Average SNR (dB) for DNN-Based MLC DeModCoder for Scenario 1 and Scenario 3.
Here, $\boldsymbol{x}$H-$\boldsymbol{y}$N in the legend represents $\boldsymbol{x}$ hidden layers and $\boldsymbol{y}$ neurons in each hidden layer used in DNN.

Considering there is no explicit computational method in DNN to calculate the optimal number of total hidden layers in the network and neurons for each layer, thus we experimentally set up them on trials and errors. In Figure 7, we show the BER for different sets of hidden layers and neurons for the proposed DNN-based MLC DeModCoder. In particular, we consider Scenario 1 (upper figure) and Scenario 3 (lower figure) to demonstrate the effect of BER based on selecting a different number of parameters for DNN. We observe for both mentioned

scenarios of conducted DeModCoder, the performance is improved with increasing the number of hidden layers and neurons (hence, decreased the BER). It is worth mentioning that the rate of improvement is higher for a smaller number of DNN parameters. It is (experimentally) noticeable that after a certain threshold value, increasing the number of layers and neurons does not significantly reduce the BER. As a result, we select the number of hidden layers and neurons in each layer based on the performance-complexity trade-off, aiming for a reasonable (BER) performance while limiting the number of parameters in NN within a certain threshold. Moreover, we also notice that the number of hidden layers and neurons we choose for one set of modulation and coding schemes might not be a good match for other modulation and coding schemes. In particular, we observe that two hidden layers and 50 neurons in each layer is a reasonably good fit for Scenario 1. On the other hand, two hidden layers with 100 neurons each show good BER performance.

Training a Single NN vs. Training Multiple NNs Over a Range of SNR

	Scenario 1		Scenario 2			Scenario 3		Scenario 4	
Variant / SNR	Single NN	Multiple NNs	Single NN	Multiple NNs		Single NN	Multiple NNs	Single NN	Multiple NNs
-10	0.31653	0.3157	0.372833	0.3725		0.284931	0.2795	0.354325	0.3163
-8	0.268308	0.2674	0.343408	0.3432		0.2462	0.2448	0.328937	0.2722
-6	0.206868	0.2062	0.302393	0.3016		0.184702	0.1832	0.293766	0.2862
-4	0.135308	0.1345	0.24801	0.2477		0.115314	0.1127	0.253123	0.2477
-2	0.066615	0.0659	0.183185	0.1827		0.048026	0.0396	0.205187	0.1521
0	0.02077	0.0204	0.110293	0.1101		0.005723	0.0032	0.134886	0.088
2	0.00208	0.0014	0.046163	0.0456		0.000847	0.00053	0.032511	0.023511
4	5.75E-05	7.00E-05	0.01466	0.0144		5.33E-05	2.53E-05	0.007033	0.005833
6	0	0	0.00179	0.0011		4.00E-06	1.00E-06	0.000102	3.2E-05
8	0	0	4.50E-05	2.50E-05		0	0	0	0
10	0	0	4.00E-06	3.00E-06		0	0	0	0

Table 1: BER vs. Average SNR (dB) for DNN-Based MLC DeModCoder Using Single and Multiple NNs for 4 Scenarios.
For each one of 4 scenarios, the comparison of BER performances for single NN and multiple NNs are presented over every even value of SNR (dB), where SNR value with a range from -10 to 10 dB.

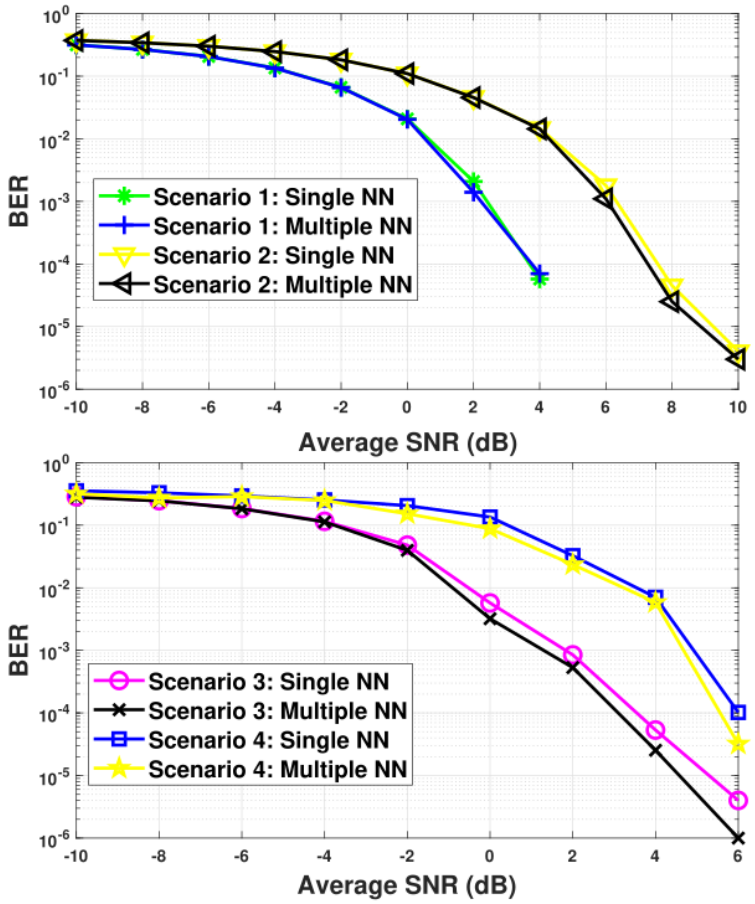

Figure 8. BER vs. Average SNR (dB) for DNN-Based MLC DeModCoder Using Single and Multiple NNs.

We display the BER for all the considered scenarios while using DNN-based MLC DeModCoder and the comparative results are presented in both Table 1 and Figure 8. We consider two different cases for training the DNN in each scenario. In one case, we consider a single NN and train it offline over a wide range of γ, whereas the other case considers $\mathbb{M}$ DNNs for $\mathbb{M}$ different values of γ. For all considered scenarios, we observe that the performance gaps between the two cases are very small. In particular, the gap between Scenario 1 and 2 is insignificant, whereas the gap is less than 0.2 dB and 0.1 dB for Scenario 3 and 4 respectively. Note that for a given scenario, we maintain the same number of hidden layers and neurons in

each layer to guarantee a fair comparison of performance. It is worth mentioning that implementing a single trained NN online requires a smaller footprint (in practical hardware setup) and thus results in lower computational complexity compared to developing multiple NNs in a device with the same form factor. Essentially, we evaluated the performances with 10^5 number of realizations, but the capability of these data might not be sufficient for calculating such a small number for the missing points. In order to obtain results beyond those data points, we were supposed to run simulations for a long time by applying a larger dataset, probably more than one day. However, we ran all the simulations on Google Colab, which provides a maximum of 12 hours runtime (Google, n.d.). Therefore, we were not able to obtain a specific result beyond a certain threshold.

Comparative Performances Among Different DeModCoders

	Scenario 1		Scenario 2		Scenario 3		Scenario 4	
Object / SNR	Proposed	Baseline1	Proposed	Baseline1	Proposed	Baseline1	Proposed	Baseline1
-10	0.31653	0.33205	0.372833	0.38067	0.284931	0.33796	0.354325	0.3163
-8	0.268308	0.29044	0.343408	0.35081	0.246201	0.29591	0.328937	0.2722
-6	0.206868	0.24131	0.302393	0.31377	0.184702	0.24168	0.293766	0.2862
-4	0.135308	0.18486	0.24801	0.26877	0.115314	0.17329	0.253123	0.2477
-2	0.06662	0.12557	0.183185	0.21529	0.048026	0.097122	0.205187	0.1521
0	0.02077	0.072817	0.110293	0.15684	0.005723	0.033661	0.134886	0.088
2	0.00208	0.03348	0.046163	0.099142	0.000847	0.005596	0.032511	0.023511
4	5.75E-05	0.01078	0.01466	0.0516	0.000054	0.0002699	0.007033	0.005833
6	0	0.002043	0.00179	0.019998	4.00E-06	1.33E-05	0.000102	3.2E-05
8	0	0.000158	4.50E-05	0.005135	0	0	0	0
10	0	5.00E-06	4.00E-06	0.000648	0	0	0	0

Table 2: BER vs. Average SNR (dB) for Proposed and Baseline Scheme 1 for 4 Scenarios. For each one of 4 scenarios, the comparison of BER performances for our proposed MLC DeModCoder and Baseline Scheme 1 are presented over every even value of SNR (dB), where SNR values with a range from -10 to 10 dB.

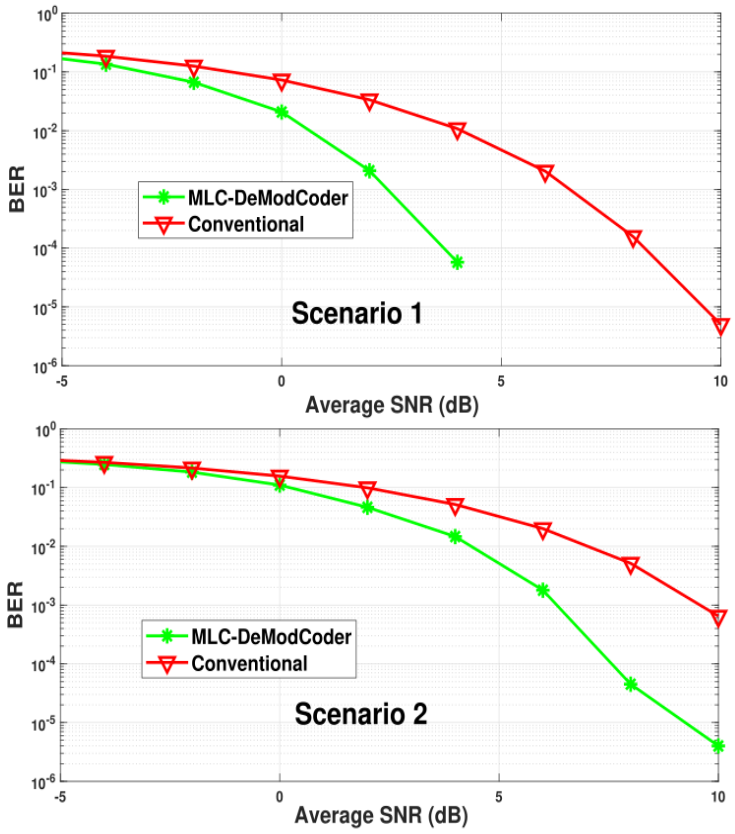

Figure 9. BER vs. Average SNR (dB) for Proposed and Baseline Scheme 1 for Scenario 1 and Scenario 2.

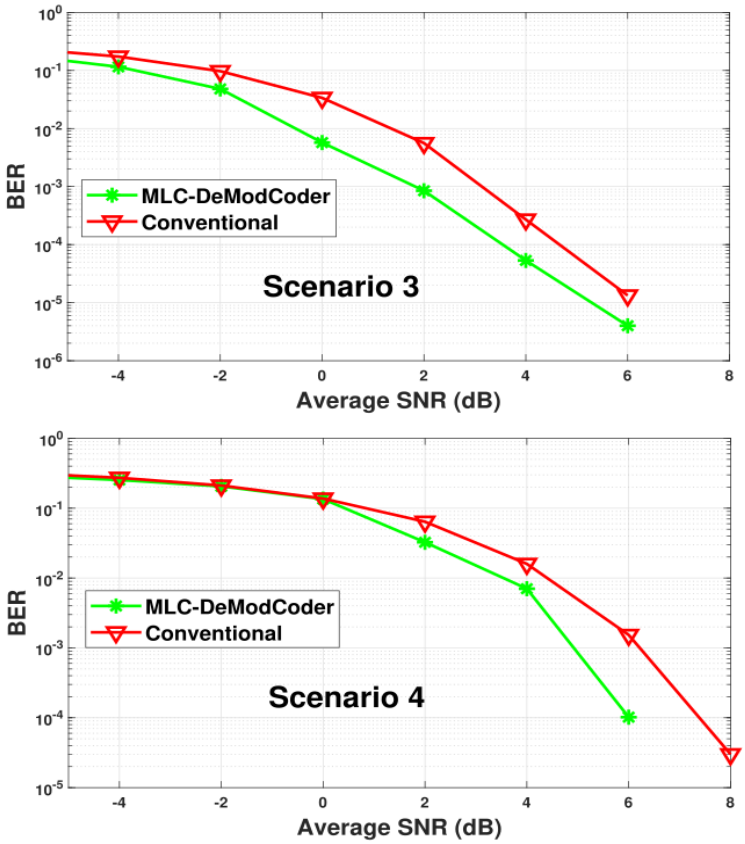

Figure 10. BER vs. Average SNR (dB) for Proposed and Baseline Scheme 1 for Scenario 3 and Scenario 4.

In Figure 9 and Figure 10, we compare the BER performances of our proposed DNN based MLC DeModCoder with baseline scheme 1, which utilize conventional baseband demodulator (BPSK for Scenario 1 and 3 and QPSK for Scenario 2 and 4) cooperating with syndrome based channel decoder for Hamming (Scenario 1 and 2) and RS coding (Scenario 3 and 4) schemes (Proakis & Salehi, 2001). Table 2 shows precise data for 4 scenarios captured from simulation results. We set two hidden layers and 50 neurons in each layer for both Scenario 1 and 2. However, in case of Scenario 3 and 4, we consider two hidden layers and 100 neurons in each layer. We observe that over a mid and high range of γ, MLC DeModCoder outperforms baseline scheme 1. The performance gap between these two manners is comparatively large for

(7,4) Hamming code than (7,3) RS code. In particular, for scenarios 1 and 2, MLC DeModCoder showed an SNR improvement of approximately 4 dB and 3.5 dB over baseline scheme 1 for Scenario 1 and 2, respectively to achieve a BER of 10^{-3}. However, for Scenario 3 and 4, these SNR improvements are roughly 1 and 1.2 dB, respectively.

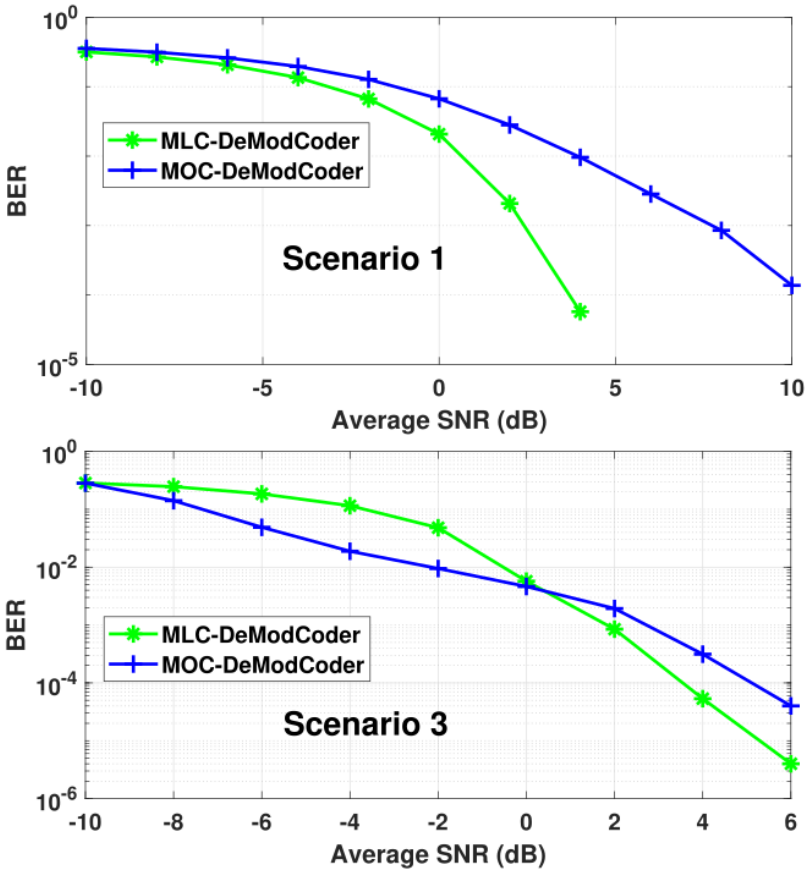

Figure 11. BER vs. Average SNR (dB) for Proposed and Baseline Scheme 2 for Scenario 1 and Scenario 3.

In Figure 11, we compare the BER performance of the proposed MLC DeModCoder with baseline scheme 2, which deploys MOC DeModCoder. Recall that MOC DeModCoder results in higher computational complexity compared to MLC DeModCoder, where the number of output nodes is $O(2^k)$ and $O(k)$ for MOC and MLC DeModCoder, respectively, k expresses the number of information bits. From the upper graph of Figure 11, we observe that for Scenario 1,

MLC DeModCoder shows around 4 dB SNR improvement in the mid and high value of γ. In case of Scenario 3, MOC performs better over low and mid values of the considered range of SNRs with approximately 4 dB SNR difference. However, MLC outperforms MOC with roughly 2 dB SNR improvement for high values of SNR.

Conclusion

In this paper, we developed a DNN-based MLC DeModCoder, which jointly demodulates and decodes the received data at the receiver. The DeModCoder can be trained offline in a supervised learning manner exploiting a training dataset that captures a wide range of SNR values. Once trained, the DNN can be deployed online without requiring the exact SNR value to be known. Note that this finding helps the system designers to avoid the additional effort of estimation of noise variance and calibrating parameters for data decoding as a single trained DNN is sufficient to perform joint demodulation and decoding over a wide range of SNRs. We have demonstrated the effectiveness of the proposed DNN based MLC DeModCoder compared to two baseline schemes by simulations. In particular, MLC DeModCoder outperforms the conventional demodulator and syndrome-based decoder over a wide range of SNRs. Furthermore, the MLC DeModCoder shows lower BER than MOC DeModCoder in the high SNR region. Note that MLC DeModCoder exhibits lower computational complexity compared to MOC DeModCoder, especially for high code rates. The research outcomes in this paper will motivate further investigations of joint demodulation and decoding for other channel coding techniques, e.g., convolutional, Turbo, Polar, LDPC codes, etc.

CHAPTER 4

JOINT BASEBAND PROCESSING WITH MULTI-LABEL CLASSIFICATION USING DEEP NEURAL NETWORKS

Background Overview

Recently, mobile communication technology has achieved gratifying accomplishments exhibiting the huge potential of development, e.g., fifth-generation (5G) mobile networks (Dahlman et al., 2018). The advancement of technology is accompanied by an increase in requirements of technical support. For instance, the capabilities of supporting multi-users and transmitting large data are fundamental and key demands for 5G technology. Multiple-input multiple-output (MIMO) plays a significant role in the application of 5G since it enables transmission of high data rate and increasing channel capacity (Kashyap & Bagga, 2014). In the MIMO system, the limitation of connection range can be avoided by overwhelmingly decreasing transmitted energy consumption and additional bandwidth demand. In general, no matter the number of transmitters, multiple receivers related to multiple channels, and hence transmitted signals are impaired by intersymbol interference, a common type of multipath fading. An equalizer is equipped at the receiver end to recuperate signals by handling these undesired influences. Moreover, a conventional communication system is block-based, where each block applies robust and optimal algorithms, e.g., baseband modulation block, channel coding block, amplifiers, equalizers, etc. However, in practice, components of each block might produce nonlinear distortions and noise to the transmitted signals. As a result, the optimal end-to-end performance may not be guaranteed. By contrast, DNN enables to deal with global tasks based on its structure and has the ability to process a large amount of data. Therefore, researchers

increasingly utilize the DL approach as assistance to develop a DNN based baseband processing system.

In the research from Ye & Li, 2017, a deep learning based joint equalization and decoding algorithm was proposed to combat channel distortions in frequency selective fading channels. It was demonstrated that the joint processing of equalization and decoding can outperform the conventional MMSE-based successive cancellation decoding approach. According to Xu et al., 2018, a cascaded equalizer and decoder were designed with CNN and DNN, respectively. In particular, a CNN equalizer compensates for the distortion of a channel while a DNN decodes the transmit bit sequence. It was demonstrated that developing CNN and DNN jointly for joint equalization and decoding shows significant performance improvements over the conventional Gaussian process for classification (combined with successive cancellation). According to Hu et al., 2019, an RNN based equalizer and decoder was designed that yields even better performance than the proposed CNN+DNN based receiver proposed by Xu et al., 2018. A preliminary study on DL architectures for channel estimation and detection in MIMO systems with low-resolution receivers has been conducted in the research by Klautau et al., 2019.

In this chapter of the thesis, we develop a joint equalizer, demodulator, and decoder (DeTecModCoder) with a fully connected DNN that applies the MLC algorithm (Brownlee, 2020; Grunau et al., 2018; M. L. Zhang & Zhou, 2006). The developed functional DeTecModCoder receives the noisy signal at its input that is impaired by different wireless fading channels and Gaussian and non-Gaussian noise and interference and hence produces data bitstream at its output. Each output label at the MLC DeTecModCoder possesses a binary class. The multiple labels at the output of the DeTechModCoder collectively represent a block of

recovered data words. This designed DNN based DeTecModCoder is applicable for flat-fading wireless channel models, linear digital modulation schemes, e.g., binary phase-shift keying (BPSK), M-ary quadrature amplitude modulator (M-QAM), etc., and linear block codes, e.g., Hamming code, Reed Solomon (RS) code, etc. Moreover, we train the functional DeModCoder block in such a way so that it shows robust performance over Gaussian and different non-Gaussian noise and interference. We leverage bit error rate (BER) as the performance metric and investigate the BER performances of the proposed receiver for a number of scenarios following the recent wireless communication standards, e.g., 5G NR, Wi-Fi 6, etc. Our objective is to reap the benefits offered by AI tools to push the boundaries of end-to-end BER performance and to alleviate the real-time computational complexity by designing a single functional block that can show robust performance in diversified harsh conditions.

The highlights of the contributions made in this chapter of the thesis are as follows:

- We develop a fully connected DNN-based single functional DeTecModCoder block that jointly combines (equalizes), demodulates, and decodes the signals received at multiple antennas for a receive-diversity system. The developed framework applies MLC to detect the data word at the output of the DNN model.
- The DNN model is trained offline with versatile datasets that incorporate different signal-to-noise ratio (SNR) values, different types of Gaussian and non-Gaussian noises, etc. Once trained, the inference model is applied in real-time for joint data detection, channel equalization (receiver combining), and baseband demodulation.
- Our trained model is robust enough to process the data for different noisy environments and a wide range of SNR values without requiring the real-time estimation of noise co-variance and statistical information of underlying noise.

The rest of the paper is organized as follows. We describe the system model and the DL-assisted DeTecModCoder in Sections 4.2 and 4.3, respectively. The simulation results are presented in Section 4.4 and the conclusions are made in Section 4.5.

System Model

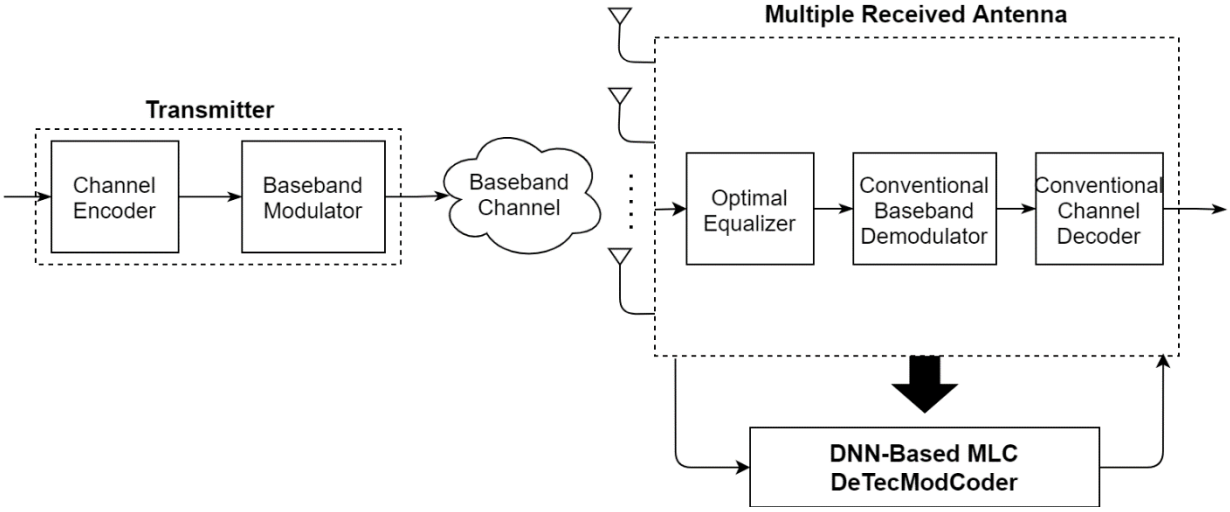

Figure 12. Multiple Receiver Communication Links with Conventional vs. AI-Based Receiver Processing.

We consider a single-input-multiple-output (SIMO) baseband communication system with a single antenna at the transmitter and multiple antennas at the receiver. In particular, D receive antennas at the receiver creates D-diversity branches, as shown in Figure 12. The transmitter is equipped with a channel encoder and a baseband modulator in a cascaded fashion. The channel encoder applies $(\tilde{n}, \tilde{k})$ linear block coding (LBC) scheme, where $\tilde{n}$ and $\tilde{k}$ represent the data-bit lengths for the codeword and the data word, respectively. We assume that the transmitter and the receiver are perfectly synchronized. In other words, the timing and frequency errors are estimated and calibrated with appropriate signal processing techniques. Let us denote channel encoding specific parameters $n = \mathcal{S}\tilde{n}$ and $k = \mathcal{S}\tilde{k}$, where $\mathcal{S}$ being an integer, represents

the number of bits used to represent an information symbol for a given channel coding scheme[4]. For instance, we set $\mathcal{T} = 1$ and $\mathcal{T} = \log_2(\tilde{n} + 1)$ for (binary) Hamming code and RS code, respectively. The transmitter sends a message r containing k information bits (e.g., $k = \tilde{k}$ for (binary) Hamming code and $k = \log_2(\tilde{n} + 1)k$ for RS code) to the receiver over a noisy channel. At first, k information bits are encoded with $(\tilde{n}, \tilde{k})$ linear block code and represented as $x = f(r)$ with n bits (e.g., $n = \tilde{n}$ for (binary) Hamming code and $n = \log_2(\tilde{n} + 1)\tilde{n}$ for RS code). Here, $f(\cdot)$ denotes the channel encoding techniques for linear block codes. The encoded bits are then passed through a baseband digital modulator, e.g., BPSK. M-QAM, etc., and hence the modulated symbols are denoted as $s = g(x)$, where $g(\cdot)$ represents the baseband modulation process. The transmitter communicates with the receiver over I equal-duration transmission time intervals.

The discrete-time complex-baseband communication model for a given time interval i can be expressed as

$$\vec{y}_i = \sqrt{\gamma}\, \vec{h}_i s_i + \vec{w}_i, \tag{3}$$

where $\vec{y}_i$, $\vec{h}_i$, and $\vec{w}_i$ denote $N \times 1$ received signal, channel gain, and noise vectors, respectively. Moreover, γ denotes the average signal-to-noise ratio (SNR) per receive antenna and s_i represents the transmitted baseband modulated symbol, which can be taken from a finite M-ary symbol alphabet $\mathcal{A}$, c.f., M-QAM, etc. In a given slot i, the d-th elements of $\vec{y}_i$, $\vec{h}_i$, and $\vec{w}_i$ are represented by $\vec{y}_{d,i}$, $\vec{h}_{d,i}$, and $\vec{w}_{d,i}$, respectively.

The small-scale flat-fading channel gain of the d-th diversity branch can be represented by $h_{d,i} \triangleq a_{d,i} e^{\theta_{d,i}}$, where magnitude $a_{d,i}$ and phase $\theta_{d,i}$ are mutually independent. $\theta_{d,i}$ is

[4] In particular, $\mathcal{S}'$-ary ($\mathcal{S} = \log_2 \mathcal{S}'$) information symbols (for the purpose of channel coding) can be constructed from Galois Field (GF) $2^{\mathcal{S}}$ (Proakis & Salehi, 2001).

uniformly distributed in $(-\pi, \pi]$ and $a_{d,i}$ can follow different (fading) distributions, e.g., Rayleigh, Rician, Nakagami-m, Nakagami-q, etc. We assume that the fading across different diversity branches can be statistically independent but are not necessarily identically distributed. The noise samples $w_{d,i}$ are independent across the diversity branches and can follow Gaussian and non-Gaussian distributions. Relevant examples of non-Gaussian noise include ϵ-mixture noise, α-stable noise, generalized Gaussian noise (GGN), CCI, etc. (Ahmed, 2014; Georgiou et al., 1999; Zhu et al., 2019).

The received signals are grouped in blocks and fed into the proposed DNN based MLC DeTecModCoder, which outputs the decoded message bits. It is worth clarifying that the designed MLC DeTecModCoder processing block jointly equalizes (detect), demodulates, and decodes the received signal in a single module (see the receiver block in Figure 12), which would otherwise be equalized, demodulated, and decoded in three consecutive (signal processing) blocks. The MLC DeTecModCoder is trained offline using a number of input datasets containing y_i, and corresponding output datasets h_i that encompasses a wide range of SNRs. We discuss the details of the training procedure in Section 4.3.

DL Assisted Joint Baseband Processing

In this section, we demonstrate the components of DL-aided MLC joint equalizer, demodulator, and decoding scheme. Recall that the DNN is trained through received noisy baseband signals and transmitted message bits as its input and output datasets, respectively.

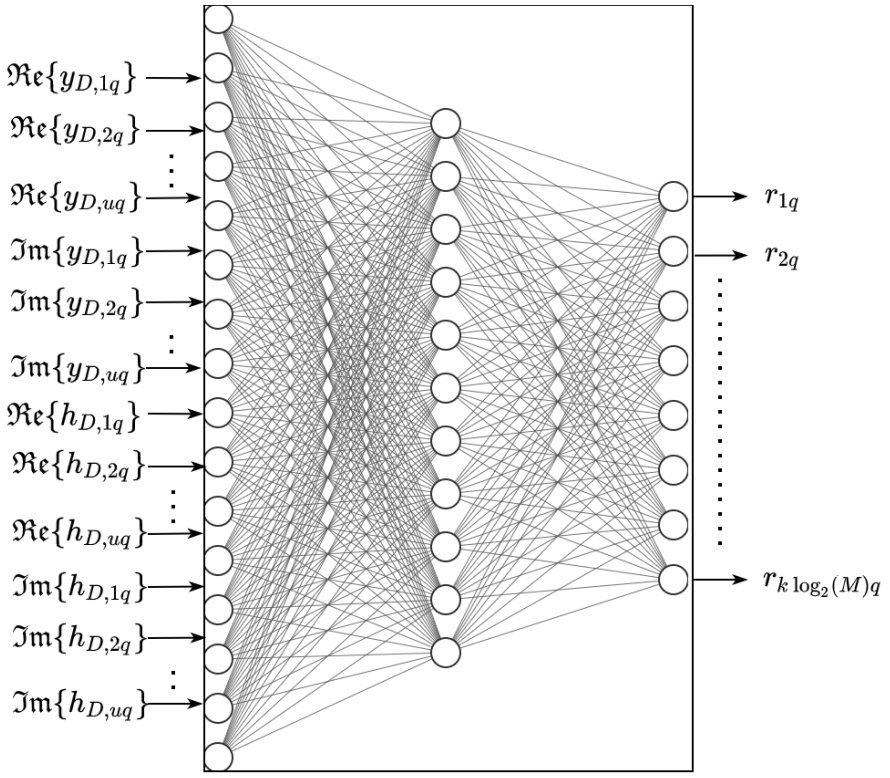

Figure 13. DNN-Based MLC DeTecModCoder.
The size of input and output training symbols for a given epoch $\boldsymbol{q}$ are $\mathbf{2 \times 2}\boldsymbol{nd}$ and $\boldsymbol{k}\,\mathbf{log_2}(\boldsymbol{M})$, respectively for $(\widetilde{\boldsymbol{n}}, \widetilde{\boldsymbol{k}})$ linear block code and $\boldsymbol{M}$-ary linear baseband modulation scheme. Note that $\boldsymbol{n} = \widetilde{\boldsymbol{n}}$ $(\boldsymbol{n} = \mathbf{log_2}(\widetilde{\boldsymbol{n}} + \mathbf{1})\,\widetilde{\boldsymbol{n}})$ and $\boldsymbol{k} = \widetilde{\boldsymbol{k}}$ $(\boldsymbol{k} = \mathbf{log_2}(\widetilde{\boldsymbol{k}} + \mathbf{1})\,\widetilde{\boldsymbol{k}})$ for Hamming code (RS code). $\mathfrak{Re}\{\cdot\}$ and $\mathfrak{Im}\{\cdot\}$ express real and imaginary parts of a complex variable.

DNN Based MLC DeTecModCoder Architecture

The proposed DNN is trained in a supervised learning manner and presented in Figure 13, where MLC is deployed at its output with multiple labels (M. L. Zhang & Zhou, 2006). Different from the common classification task, where a single label correlates with multiple classes, the output labels of MLC are not mutually exclusive. In addition, for the MLC method, the number of output nodes in a DNN can be nominated as the number of targeted labels by the structural characteristic of DNN (Brownlee, 2020a). In this case, the output nodes of the DNN can combinedly express a sequence of binary symbols.

Training Datasets Configuration: In order to gather training datasets of DNN-based MLC DeTecModCoder, we assume that within the training process, every epoch involves P batches, and the total number of epochs is Q. In particular, the complete training datasets are applied for a single epoch when $P = 1$. Assuming that a set of message bits sequence generated by the transmitter is regarded as $\mathcal{R}_q$ in a given epoch $q \in \{1,2,\cdots,Q\}$. These data sequences are progressively passed channel coding block and baseband modulation block with mapped bitstreams $\mathcal{X}_q$ and $\mathcal{S}_q$, $q \in \{1,2,\cdots,Q\}$, respectively. Following equation (3), the modulated signals are impaired with AWGN, GNN, or ϵ-miture noise. Then the noisy signals $\mathcal{Y}_q$ are transmitted through D number of diverse branches, cooperated with channel gain $\mathcal{H}_{d,q}$, and hence our received signals are represented as $\mathbb{Y}_{d,q}$, $q \in \{1,2,\cdots,Q\}$, $d \in \{1,2,\cdots,D\}$. The sets of digital signals $\{\mathcal{Y}_{1,1}, \mathcal{Y}_{1,2}, \cdots, \mathcal{Y}_{D,Q}, \mathcal{H}_{1,1}, \mathcal{H}_{1,2}, \cdots, \mathcal{H}_{D,Q}\}$ and $\{\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_Q\}$ are arranged as our training input and output datasets, respectively.

Note that the input data sequence $\mathbb{Y}_{d,q}$, $d \in \{1,2,\cdots,D\}$ and $q \in \{1,2,\cdots,Q\}$, consists of real parts $\Re\mathrm{e}\{x\}$ and imaginary parts $\Im\mathrm{m}\{x\}$ for a complex variable x, and hence $\mathbb{Y}_{d,q}$ can be denoted as

$\left[\Re\mathrm{e}\{y_{D,1q}\}, \Re\mathrm{e}\{y_{D,2q}\}, \cdots, \Re\mathrm{e}\{y_{D,uq}\}, \Im\mathrm{m}\{y_{D,1q}\}, \Im\mathrm{m}\{y_{D,2q}\}, \cdots, \Im\mathrm{m}\{y_{D,uq}\}, \Re\mathrm{e}\{h_{D,1q}\}, \Re\mathrm{e}\{h_{D,2q}\}, \cdots, \Re\mathrm{e}\{h_{D,uq}\}, \Im\mathrm{m}\{h_{D,1q}\}, \Im\mathrm{m}\{h_{D,2q}\}, \cdots, \Im\mathrm{m}\{h_{D,uq}\}\right]^T$, where $y_{d,zq}$ represents the sample of received signals and $h_{d,qz}$ expresses the channel gain, $d \in \{1,2,\cdots,D\}$, $z \in \{1,2,\cdots,n\}$ and $q \in \{1,2,\cdots,Q\}$. The considered output training sets $\mathcal{R}_q$, $q \in \{1,2,\cdots,Q\}$, has a similar composition $\left[r_{1q}, r_{2q}, \cdots, r_{k\log_2(M)q}\right]^T$, where h_{zq} denotes the sample of transmitted data streams, $z \in \{1,2,\cdots,n\}$ and $q \in \{1,2,\cdots,Q\}$. It is worth noting that the number of data bits for $\mathbb{Y}_{d,q}$ and $\mathcal{R}_q$ are $2 \times 2nd$ and $k\log_2(M)$, respectively. Specifically, $y_{d,zq}$ is a complex number comprising

real and imaginary parts, as well as the signals are in d-th diversity branch, thus the combination of received signals has $2nD$ data bits. In addition, the flat-fading channel gain $h_{d,zq}$ has the same size of $y_{d,zq}$. By jointly combining received signals with channel gain, we obtain our input training sets with the size $2 \times 2nD$. For instance, assuming that a set of message bits is processed by (7,4) Hamming code with BPSK modulator to a 2-received antennas system, $\mathbb{Y}_{d,q}$ and $\mathcal{R}_q$ consist of 4 and 56 bits, respectively. Therefore, we equip $2 \times 2nd$ nodes at input layer and $k \log_2(M)$ output labels at the output layer for a fully connected DNN to deploy a DNN-based MLC DeTecModCoder.

Parameters of DNN: Within a given hidden layer $l \in \{1,2,\cdots,\mathcal{L}\}$, an MLC-based DNN can be defined as $\vec{\delta_l} = \{\xi_{1l}, \xi_{2l}, \cdots, \xi_{\mathcal{N}l}, \mu_{1l}, \mu_{2l}, \cdots, \mu_{\mathcal{N}l}\}$, where ξ and μ represent weight and bias for each neuron, respectively. Here, $\mathcal{L}$ is the total number of hidden layers and each layer contains $\mathcal{N}$ number of neurons. We feed input and output training datasets $\mathbb{Y}_{d,q}$ and $\mathcal{R}_q$ to train the MLC DeTecModCoder in order to optimize $\vec{\delta} = \{\delta_1, \delta_2, \cdots, \delta_{\mathcal{L}}\}$ as the following equation:

$$\delta^* = \arg\min G(\mathcal{R}_q, \mathcal{R'}_q), \tag{4}$$

where G denotes a function of binary cross-entropy loss (Grunau et al., 2018), and $\mathcal{R'}_q$ gives the approximation of output sequence $\mathcal{R}_q$. Within hidden layers, we apply the ReLU activation function for most of the realizations; however, we select the PReLU activation function instead for the realizations with 'dying ReLU' (Brownlee, 2019; K. He et al., 2015). At the output layer of MLC DeTecModCoder, we deploy sigmoid activation function (Ramachandran et al., 2018) and optimize function (4); meanwhile, stochastic gradient descent manner (e.g. Adam optimization, etc.) and backpropagation algorithms are used in training the DNN model (Goodfellow et al., 2016).

Arrangement of Interference Model: Recall that the proposed DNN-based MLC DeTecModCoder is trained offline and then applied in real-time to jointly combine equalization, demodulation, and decoding blocks. Based on this structure, we feed received signals $\{y_{1,i}, y_{1,i+1}, \cdots, y_{1,i+n-1}, \cdots, y_{D,i}, y_{D,i+1}, \cdots, y_{D,i+n-1}, h_{1,i}, h_{1,i+1}, \cdots, h_{1,i+n-1}, \cdots, h_{D,i}, h_{D,i+1}, \cdots, h_{D,i+n-1}\}$ as the input of the MLC DeTecModCoder generating $\{r'_1, r'_2, \cdots, r'_{k\log_2(M)}\}$ as the output, where r'_s expresses the decoded data bit corresponded to the transmitted information bit r_s, $s \in \{1,2,\cdots, k\log_2(M)\}$.

Patterns of MLC DeTecModCoder

Considering the training method for DNN with its flexible structure, we tend to make some variations for the proposed MLC DeTecModCoder before deploying them online. Following are three different mentioned variants.

1) MLC Pattern 1 - MLC DeTecModCoder with Single NN: In this pattern, we utilize a large dataset to train only a single DNN. Here, both input and output datasets over a wide range of SNR values γ compose the large training datasets.
2) MLC Pattern 2 - MLC DeTecModCoder with Low Diversity NN: In this pattern, we evenly divide γ into three ranges of SNR values. For each range of divided γ, we train one DNN using MLC, and hence we finally obtain three trained MLC-based DNNs.
3) MLC Pattern 3 - MLC DeTecModCoder with Multiple NN: In this pattern, we train an MLC-based DNN for every SNR value. In other words, let us assume that γ contains $\mathbb{V}$ number of SNR values, and hence we train $\mathbb{V}$ number of DNNs individually.

Baseline Scheme

For the purpose of evaluating performances for conducted MLC DeTecModCoder, we arrange a baseline scheme based on conventional algorithms. Within this baseline scheme, we

consider the fundamental baseband processing blocks, including the optimal maximum likelihood equalization algorithm, conventional baseband demodulation, and syndrome-based channel decoding algorithm (Proakis & Salehi, 2001). These conventional signal processing algorithms are robust; thus, they are recognized as a reliable standard to evaluate the performances of DNN.

Simulation Result

In this section, the BER performances obtained by the designed DNN-based MLC DeTecModCoder are evaluated for joint processing of equalization, demodulation, and channel decoding, and compared with the two arranged baseline schemes. In our experimental module, we utilize linear block codes for channel coding techniques, containing Hamming code and RS code, especially (7,4) Hamming code and (7,3) RS code, and linear digital modulation techniques, including BPSK and 4-QAM. It is worth noting that the implemented research can be developed and extended to other advanced linear block codes, such as low-density parity check (LDPC) codes, etc. and baseband modulation techniques with higher order, such as 64-QAM, 256-QAM, 1024-QAM, etc. Moreover, we investigate the performance of joint DeTecModCoder for MIMO cases, especially for 2 receive antennas (Rx2) and 4 receive antennas (Rx4). Combining baseband modulation and channel coding, we consider the following for scenarios for both Rx2 and Rx4.

- Scenario 1: (7,4) Hamming code + BPSK modulation
- Scenario 2: (7,4) Hamming code + QPSK modulation

In order that our design is based on DNN-based MLC DeTecModCoders, we apply Pandas and NumPy libraries in Python framework with Tensorflow/Keras modules to design the simulation platform (Abadi et al., 2016). It is vital to develop common benchmarks and open

datasets in order to compare the performances of different AI tool-based algorithms. While this is a standard concept in other technical fields, it is not common in wireless communication networks. The reason behind this different trend is that the communication systems involve man-made signals that can be precisely generated artificially in a Monte-Carlo simulation framework while adhering to communication standards and protocols.

We consider two different scenarios of multiple-output antenna cases, two sets of 10^5 realizations including random data bits and noise samplings are generated as experimental group and control group, respectively. In order to evaluate performance, we generate another collection of 10^5 independent realizations containing random data bits and noise samplings for non-NN-based demodulator and decoder. In particular, practical training datasets can be gathered through field experiments from live networks or through controlled environments from laboratories. Therefore, we can utilize the practical datasets to train the NN.

Training Different Diversity NNs Over a Range of SNR

	Scenario 1				Scenario 2		
Variant / SNR	Single NN	Low Diversity NNs	Multiple NNs		Single NN	Low Diversity NNs	Multiple NNs
-10	0.256805	0.2524075	0.25018		0.326065	0.304525	0.2148
-8	0.2097575	0.2068075	0.20973		0.2901425	0.272955	0.1917625
-6	0.162055	0.16175	0.164715		0.248605	0.24097	0.16273
-4	0.118465	0.12172	0.11806		0.2034025	0.20951	0.13007
-2	0.0823975	0.076785	0.0754875		0.157465	0.1495075	0.086115
0	0.0565675	0.0435425	0.0412625		0.11304	0.1013875	0.0448875
2	0.0392975	0.0276975	0.0139675		0.0751975	0.065945	0.012275
4	0.0287225	0.0201675	2.35E-03		0.04558	0.045275	0.0035175
6	0.0228625	0.0011875	0.000455		0.0259975	0.013355	0.00121
8	0.021115	0.0005525	0.00019		0.015325	0.0038025	9.30E-04

Table 3: BER vs. Average SNR (dB) for Scenario 1 and Scenario 2 Using Single, Low Diversity and Multiple NNs for Rx2.
For scenario 1 and scenario 2 at Rx2, the comparison of BER performances for Single NN, Low Diversity NNs, and Multiple NNs are presented over every even value of SNR (dB), where SNR values with a range from -10 to 8 dB.

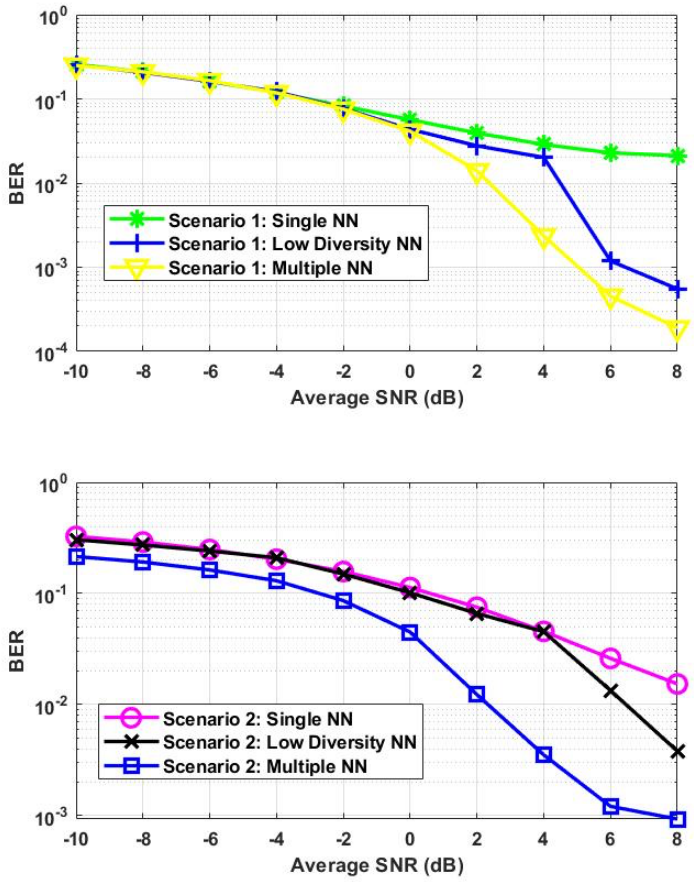

Figure 14. BER vs. Average SNR (dB) for Scenario 1 and Scenario 2 Using Single, Low Diversity and Multiple NNs for Rx2.

In Figure 14, we present the BER performance for Scenario 1 and Scenario 2 using DNN-based MLC DeTecModCoder, and the corresponding data are listed in Table 3. For each scenario, we consider three different cases to train the DNN. In the first case, we create a single NN and train it offline over a wide range of γ. In the second case, we generate three NNs, where each NN is trained over $\frac{1}{3}$ total values of γ. For the last case, $\mathbb{V}$ number of NNs are utilized for $\mathbb{V}$ different γ values. For both Scenarios 1 and 2, we observe that increasing the number of NNs, the performances show an upward trend; that is, BER decreases with compressing diversity in training symbols. In practice, for low values of SNR, three diversity receiver patterns show similar BER. However, comparing the BER performance from single NN to low diversity NN

cases for a high range of SNR values, the BER improves approximately from 10^{-2} to 10^{-3} and from $10^{-1.5}$ to $10^{-1.8}$ for Scenario 1 and Scenario 2, respectively. From low diversity NN to multiple NN cases, the BER performances present the rough improvements from 10^{-3} to $10^{-3.5}$ and from 10^{-2} to 10^{-3}, respectively. In order to ensure the accuracy of simulation results, we used the same number of hidden layers and neurons in each layer for a given scenario. In particular, the results for multiple NNs were showed are just for demonstration purposes. That is true that if we apply multiple NNs, there is a requirement for exact SNR values to be known. But our purpose of showing BER performances of multiple NNs was just to compare the proposed algorithm using one single NN. The results here are different from the simulation result in Chapter 3. In the comparison outcomes of Chapter 3, when we have not considered channel fading and equipped equalizer, the result of training one single NN and multiple NNs presented similar performance. By contrast, as shown in Table 3 and Figure 14, when the complexity of the system increased by considering fading channel and applying equalizer to combat the impairment caused by channels, the BER performance captured from one single NN and multiple NNs showed a large gap and did not close as result in Chapter 3.

Performances Comparison Under Different Noises Among Different DeTecModCoders

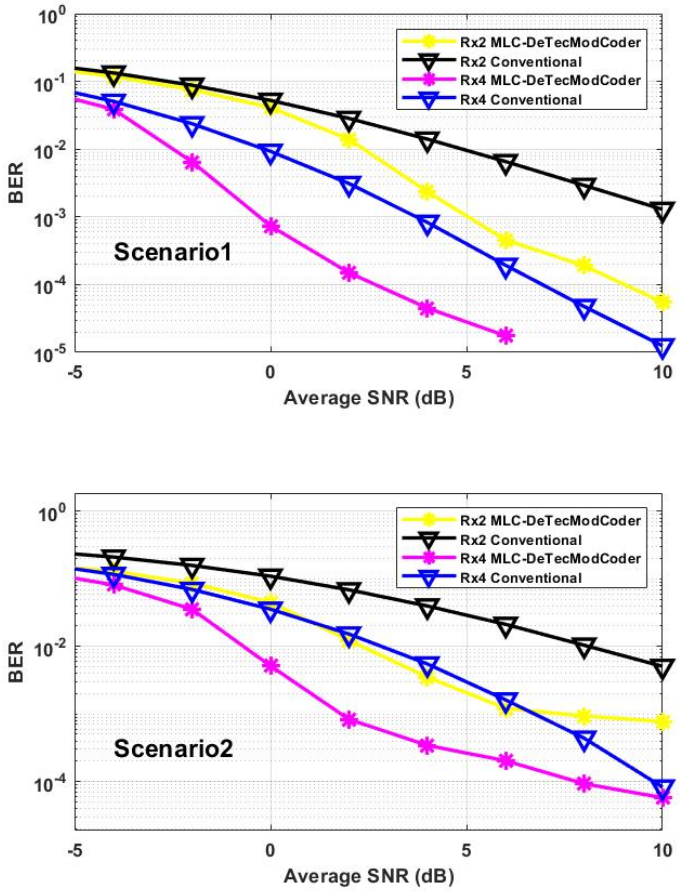

Figure 15. BER vs. Average SNR (dB) for The Proposed and Baseline Scheme for Scenario 1 and Scenario 2 under AWGN for Rx2 and Rx4.

In Figure 15, we compare the BER performances of the proposed DNN-based MLC DeTecModCoder with the baseline scheme under AWGN for both Rx2 and Rx4 cases. Recall that in the baseline scheme, we apply optimal maximum likelihood equalizer, conventional baseband modulator (BPSK for Scenario 1 and QPSK for Scenario 2), and syndrome-based channel decoder for Hamming code (Proakis & Salehi, 2001). In case of Scenario 1 for all noise configurations, we consider 2 hidden layers and 100 neurons in each layer with 500 epochs and 64 batch sizes for Rx2; whereas 2 hidden layers and 150 neurons in each layer with 500 epochs

and 64 batch size for Rx4. In case of Scenarios 2 for all noise configurations, we arrange to consider 2 hidden layers and 100 neurons in each layer with 500 epochs for Rx2; whereas 2 hidden layers and 150 neurons in each layer with 1000 epochs and 128 batch size for Rx4. We observe that MLC DeTecModCoder under AWGN outperforms the baseline scheme over a mid and high range of γ. To achieve BER of 10^{-3}, the performance gaps exhibit approximately 5 dB and 4 dB of SNR improvements over the baseline scheme for Scenario 1 at Rx2 and Rx4, respectively. However, these SNR improvements show approximately 5 dB for Scenario 2 at both Rx2 and Rx4. This finding indicates that training a single NN over a wide range of SNR values is enough for real-time data recovery. Essentially, deploying a single trained NN online have much lower computational complexity without the requirement of exact SNR values.

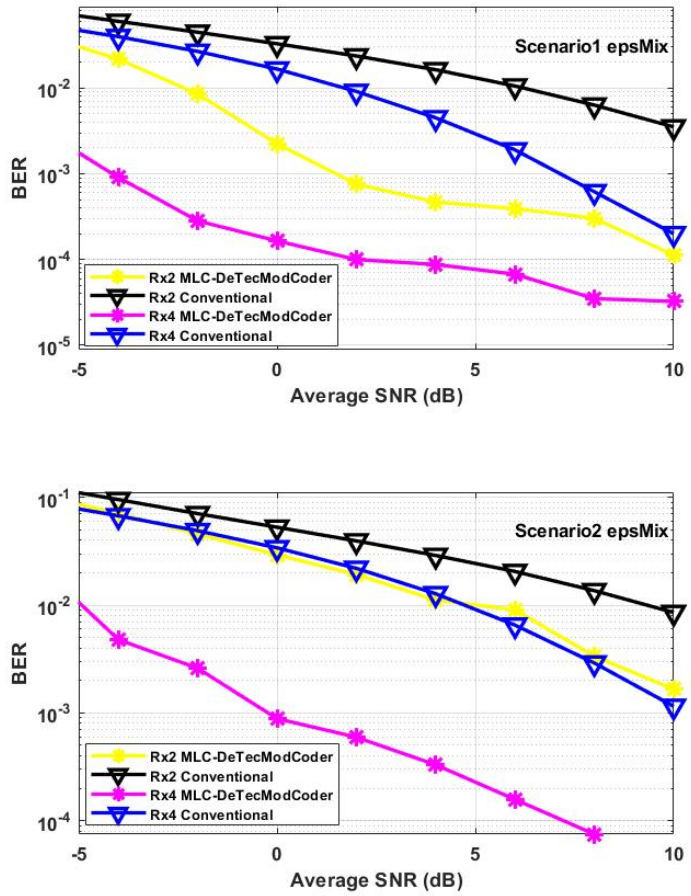

Figure 16. BER vs. Average SNR (dB) for The Proposed and Baseline Scheme for Scenario 1 and Scenario 2 under ϵ-Mixture Noise for Rx2 and Rx4.

In Figure 16, we compare the BER performances of the proposed DNN-based MLC DeTecModCoder with the baseline scheme under ϵ-mixture noise for both Rx2 and Rx4 cases. The parameters of NN that we utilized here are the same as in the AWGN case. We observe the performance gaps are comparatively larger than AWGN configurations. To achieve BER of $10^{-2.5}$, the SNR improvements present approximately 10 dB over baseline scheme for Scenario 1 at both Rx2 and Rx4; while showing roughly 4 dB and 8 dB for Scenario 2 in order to achieve BER of 10^{-2} at Rx2 and Rx4, respectively.

	Scenario 1				Scenario 2			
Object / SNR	Rx2 Proposed	Rx2 Convention	Rx4 Proposed	Rx4 Convention	Rx2 Proposed	Rx2 Convention	Rx4 Proposed	Rx4 Convention
-10	0.224895	0.25749	0.2873725	0.31731	0.15093	0.18245	0.2106575	0.25877
-8	0.1848525	0.21129	0.253445	0.27746	0.105	0.1306	0.164875	0.20915
-6	0.14083	0.1637	0.21126	0.23316	0.06322	0.084733	0.12005	0.15679
-4	0.1013175	0.11858	0.17003	0.1861	0.030285	0.048325	0.07463	0.10733
-2	0.065615	0.07984	0.1287725	0.13966	0.0058325	0.024753	0.0337675	0.066208
0	0.03549	0.050197	0.0890225	0.09781	0.001015	0.011033	0.00355	0.036135
2	0.0147225	0.028795	0.0562325	0.063387	0.0002875	0.00419	0.0011925	0.016815
4	0.00236	0.015595	0.029975	0.038055	7.75E-05	0.001325	0.000425	0.006985
6	0.000845	0.007875	0.012835	0.02156	4.50E-05	0.0003675	0.0001	0.0024
8	0.0003175	0.003905	0.002855	0.011468	0	8.75E-05	0.0001525	0.0006925
10	0.000195	0.0017325	0.001255	0.005665	0	1.00E-05	0	0.000185

Table 4: BER vs. Average SNR (dB) for The Proposed and Baseline Scheme for Scenario 1 and Scenario 2 under GGN for Rx2 and Rx4.

For scenario 1 and scenario 2 at Rx2 and Rx4, the comparison of BER under GGN has presented performances for proposed DeTecModCoder and conventional algorithm over every even value of SNR (dB), where SNR values with a range from -10 to 10 dB.

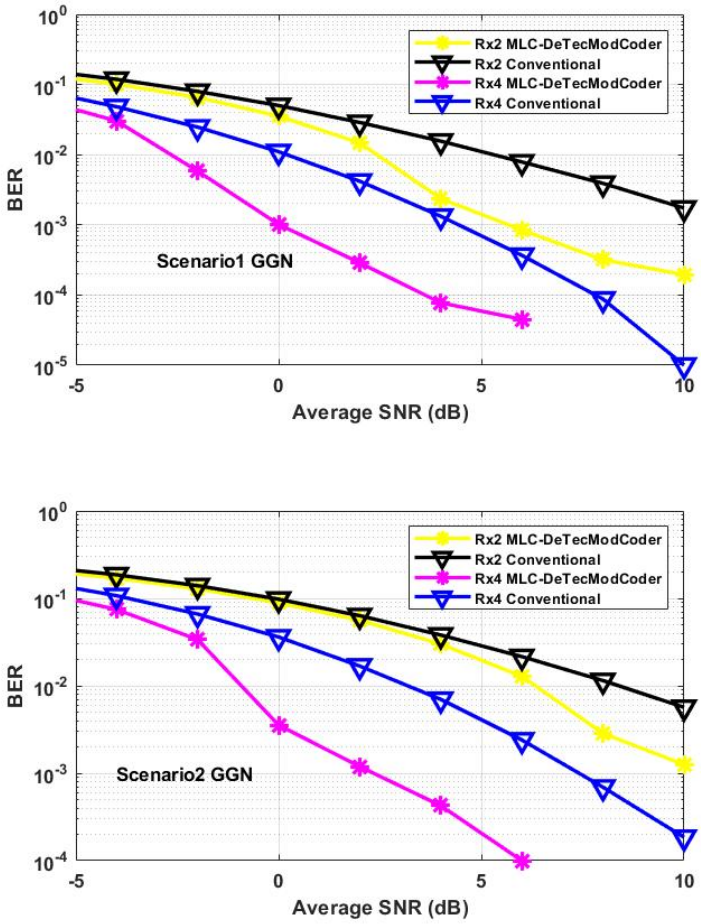

Figure 17. BER vs. Average SNR (dB) for The Proposed and Baseline Scheme for Scenario 1 and Scenario 2 under GGN for Rx2 and Rx4.

In Figure 17, we compare the BER performances of the MLC DeTecModCoder with the baseline scheme under GNN for both Rx2 and Rx4 cases. The figure is generated through the data sequences in Table 4. In this case, we are using the same parameters of NN as the AWGN case. Similar to the results for ϵ-mixture noise, MLC DeTecModCoder under GGN also outperforms the baseline scheme over a mid and high range of γ. The difference is that the performance gaps here are not that large. In order to reach 10^{-2} BER value, for Scenario 1, the SNR improves roughly 2 dB and 2.5 dB for Rx2 and Rx4, respectively; by contrast for Scenario 2, the increasing SNR is around 2 dB for Rx2 and 3.5 dB for Rx4.

Conclusion

In this chapter, we developed a DNN-based MLC DeTecModCoder at the received end that jointly equalizes, demodulates, and decodes the received data bits. We trained this MLC DeTecModCoder offline by applying a supervised learning approach over a versatile large number of SNR values. This trained model can be deployed in real-time applications without the exact information of underlying noise. We considered a baseline scheme to evaluate the performance of the proposed DNN model under different types of noise and channel fading, and hence the proposed model is robust enough to process the data in different practical environment conditions. We demonstrated the effectiveness of the developed MLC DeTecModCoder by representing the SNR improvements compared to the conventional schemes. Moreover, we indicated that in order to increase performance (by reducing BER), it is adoptable to decrease the diversity in training symbols. The outcomes of this chapter motivated us to further attempt more complicated baseband processing techniques jointly, e.g., MMSE equalizer, Turbo, LDPC codes, etc.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Conclusion

In this thesis, we proposed DNN-based MLC receiver designs, which jointly combined partial signal processing blocks at the received end, and developed them for both single and multiple antenna cases. By the setup of versatile diversity branches at the received end, we indicated that diminishing the diversity of training symbols can be helpful to reduce the BER. The objective of applying the MLC approach at the output layer was that compared to the MOC approach, the computational complexity was decreased from $O(2^k)$ to $O(k)$, where k is the number of information bits. Furthermore, the design model was trained offline in a supervised learning approach through a training dataset gathered over a wide range of SNRs, various Gaussian and non-Gaussian noise, and channel fading. Once trained, the DNN model was deployed in real-time signal processing without knowing SNR values. This finding showed that our proposed DNN-based MLC receiver design model has the ability of baseband processing in noisy conditions without the requirements of accurate information for underlying noise and SNR values. Simulation results demonstrated that BER performances generated from developed intelligent schemes outperformed those from the corresponding conventional schemes with approximately 3 dB SNR improvements.

Future Works

Although our current design presents efficient performance compared to several conventional block-based systems, we observed that only applying fully connected DNN may not guarantee the robust performance provided by more complicated and accurate conventional algorithms. In addition, the statistical datasets we applied are not practical data despite the

simulation data and practical data are close enough. Thus, we plan to extend our idea of AI-aided MLC baseband design in following potential directions incorporating timing and frequency errors. First, instead of linear block codes, we could apply more advanced channel coding techniques, e.g., convolutional, Polar, Turbo codes, etc. Second, in this research, we utilized conventional approaches of encoding and modulation, and focus on receiver design. The further investigation could be a joint transmitter and receiver design; that is, a framework of autoencoder-based joint transmitter and receiver, and hence we could have more control on transmission process. Third, we could explore the application of the MLC approach to solving other problems in the PHY design, e.g., digital precoding, etc., as well as in modulation classification problems, e.g., user grouping, antenna selection, etc.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Research, G. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *ArXiv*, *abs/1603.0*. http://arxiv.org/abs/1603.04467

Agiwal, M., Roy, A., & Saxena, N. (2016). Next generation 5G wireless networks : A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *18*(3), 1617–1655. https://doi.org/10.1109/COMST.2016.2532458

Ahmed, I. (2014). *Resource allocation in wireless systems with conventional and energy harvesting nodes* [Doctoral dissertation, University of British Columbia]. Vancouver : University of British Columbia Library. https://doi.org/10.14288/1.0167609

Ahmed, I., & Allen, E. J. (2020). Deep learning based diversity combining for generic noise and interference. *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 1–4. https://doi.org/10.1109/VTC2020-Spring48590.2020.9129375

Akın, S., Penner, M., & Peissig, J. (2020). Joint channel estimation and data decoding using SVM-based receivers. *ArXiv, abs/2012.02523.*

Aldossari, S. M., & Chen, K. C. (2019). Machine learning for wireless communication channel modeling: An overview. *Wireless Personal Communications*, *106*(1), 41–70. https://doi.org/10.1007/s11277-019-06275-4

Bakarola, V. (2021). *Eight commands that every Colab user should know*. Medium. https://medium.com/analytics-vidhya/the-google-colab-system-specification-check-69d159597417

Bhaskar, A. (2020, September 12). *Reed Solomon codes|Encoding and decoding| Applications and advantages|Information theory and coding* [Video]. Youtube. https://www.youtube.com/watch?v=KAZA6iVC23M

Brownlee, J. (2019). *A gentle introduction to the rectified linear unit (ReLU)*. Machine Learning Mastery. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Brownlee, J. (2020a). *Multi-label classification with deep learning*. https://machinelearningmastery.com/multi-label-classification-with-deep-learning/

Brownlee, J. (2020b). *Supervised and unsupervised machine learning algorithms*. Machine Learning Mastery. https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

Chamier, L. von, Jukkala, J., Spahn, C., Lerche, M., Hernández-pérez, S., Mattila, P., Karinou, E., Holden, S., Solak, A. C., Krull, A., Buchholz, T.-O., Jug, F., Royer, L. A., Heilemann,

M., Laine, R. F., Jacquemet, G., & Henriques, R. (2020). ZeroCostDL4Mic: An open platform to simplify access and use of deep-learning in microscopy. *Nature Communications*. https://doi.org/10.1038/s41467-021-22518-0

Chan, V. W. S. (1997). Chapter 3 coding and error correction in optical fiber communications systems. *Optical fiber telecommunications IIIA*, (pp. 42–62). Academic Press. https://doi.org/10.1016/B978-0-08-051316-4.50007-2

Clerckx, B., & Oestges, C. (2013). Chapter 3 analytical MIMO channel representations for system design. *Mimo wireless networks* (pp. 59–100). Academic Press. https://doi.org/10.1016/b978-0-12-385055-3.00003-1

Google. (n.d.). *Colaboratory Frequently Asked Questions*. https://research.google.com/colaboratory/faq.html#gpu-availability

Costello, D. J., & Forney, G. D. (2007). Channel coding: The road to channel capacity. *Proceedings of the IEEE*, *95*(6), 1150–1177. https://doi.org/10.1109/JPROC.2007.895188

Dahlman, E., Parkvall, S., & Sköld, J. (2018). *5G NR: The Next Generation Wireless Access Technology*. Academic Press.

Djordjevic, I. (2012). Chapter 6 classical error correcting codes. *Quantum information processing and quantum error correction* (pp. 175–225). Academic Press. https://doi.org/10.1016/B978-0-12-385491-9.00006-X

Farsad, N., & Goldsmith, A. (2017). Detection algorithms for communication systems using deep learning. *ArXiv, abs/1705.08044*.

Faruque, S. (2016). Introduction to channel coding. *Radio frequency channel coding made easy* (pp. 1–16). Springer, Cham. https://doi.org/10.1007/978-3-319-21170-1_1

Gallion, P. (2016). 3 - Basics of incoherent and coherent digital optical communications. *Undersea fiber communication systems* (2nd ed., pp. 55–117). Academic Press. https://doi.org/10.1016/B978-0-12-804269-4.00003-9

Garg, V. (2007). Chapter 8 speech coding and channel coding. *Wireless communications & networking* (pp. 215–248). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-373580-5.50042-9

GeeksforGeeks. (2020a). *Activation functions in neural networks.* https://www.geeksforgeeks.org/activation-functions-neural-networks/

GeeksforGeeks. (2020b). *Layers of OSI model.* https://www.geeksforgeeks.org/layers-of-osi-model/

Georgiou, P. G., Tsakalides, P., & Kyriakakis, C. (1999). Alpha-stable modeling of noise and robust time-delay estimation in the presence of impulsive noise. *IEEE Transactions on Multimedia*, *1*(3), 291–301. https://doi.org/10.1109/6046.784467

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org

Grami, A. (2016). Chapter 10 error-control coding. *Introduction to digital communications* (pp. 409–455). Academic Press. https://doi.org/10.1016/B978-0-12-407682-2.00010-7

Grunau, S., Block, D., & Meier, U. (2018). Multi-Label Wireless Interference Identification with Convolutional Neural Networks. *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 187-192.

Haque, A. U., Saeed, M., & Siddiqui, F. A. (2012). Comparative study of BPSK and QPSK for wireless networks over NS2. *International Journal of Computer Applications*, *41*(19), 8–12. https://doi.org/10.5120/5647-7809

Harkous, H., & Aberer, K. (2017). If you can't beat them, join them": A usability approach to interdependent privacy in cloud apps. *CODASPY 2017 - Proceedings of the 7th ACM Conference on Data and Application Security and Privacy*, 127–138. https://doi.org/10.1145/3029806.3029837

He, H., Jin, S., Wen, C. K., Gao, F., Li, G. Y., & Xu, Z. (2019). Model-driven deep learning for physical layer communications. *IEEE Wireless Communications*, *26*(5), 77–83. https://doi.org/10.1109/MWC.2019.1800447

He, H., Wen, C. K., Jin, S., & Li, G. Y. (2018). Deep learning-based channel estimation for beamspace mmWave massive MIMO systems. *IEEE Wireless Communications Letters*, *7*(5), 852–855. https://doi.org/10.1109/LWC.2018.2832128

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026-1034.

He, Y., Jiang, M., Ling, X., & Zhao, C. (2020). Robust BICM design for the LDPC coded DCO-OFDM: A deep learning approach. *IEEE Transactions on Communications*, *68*(2), 713–727. https://doi.org/10.1109/TCOMM.2019.2954399

Hu, Y., Zhao, L., & Hu, Y. (2019). Joint channel equalization and decoding with one recurrent neural network. *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB*, 1–4. https://doi.org/10.1109/BMSB47279.2019.8971938

Huang, H., Guo, S., Gui, G., Yang, Z., Zhang, J., Sari, H., & Adachi, F. (2019). Deep learning for physical-layer 5G wireless techniques: Opportunities, challenges and solutions. *IEEE Wireless Communications*, *27*(1), 214–222. https://doi.org/10.1109/MWC.2019.1900027

Kashyap, R., & Bagga, J. (2014). Equalization techniques for MIMO systems in wireless communication : A review. *International Journal of Engineering and Advanced Technology (IJEAT)*, *3*(5), 260–264.

Klautau, A., Gonzalez-Prelcic, N., Mezghani, A., & Heath, R. W. (2018). Detection and channel

equalization with deep learning for low resolution MIMO systems. *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 1836–1840. https://doi.org/10.1109/ACSSC.2018.8645551

Klima, R., Sigmon, N., & Stitzinger, E. (2020). Reed-Solomon codes. *Applied Abstract Algebra with Maple™ and Matlab®*, *1*(3), 137–172. https://doi.org/10.1201/b19010-5

Long, M. (2014). Sound Reinforcement Systems. *Architectural acoustics* (2nd ed., pp. 673–721). Academic Press. https://doi.org/10.1016/b978-0-12-398258-2.00018-0

Lopez-Gordo, M., & Pelayo, F. (2013). A binary phase-shift keying receiver for the detection of attention to human speech. *International Journal of Neural Systems*, *23*(4). https://doi.org/10.1142/S0129065713500160

Luaces, O., Díez, J., Barranquero, J., del Coz, J. J., & Bahamonde, A. (2012). Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, *1*(4), 303–313. https://doi.org/10.1007/s13748-012-0030-x

Ly, A., & Yao, Y.-D. (2021). A review of deep learning in 5G research: Channel coding, massive MIMO, multiple access, resource allocation, and network security. *IEEE Open Journal of the Communications Society*, *2*(1), 396–408. https://doi.org/10.1109/ojcoms.2021.3058353

Mahmood, A., Bennamoun, M., An, S., Sohel, F., Boussaid, F., Hovey, R., Kendrick, G., & Fisher, R. B. (2017). Chapter 21-deep learning for coral classification. *Handbook of neural computation* (1st ed.). Elsevier Inc. https://doi.org/10.1016/B978-0-12-811318-9.00021-1

Marcus, G. (2018). Deep learning: A critical appraisal. *ArXiv*. http://arxiv.org/abs/1801.00631

Mitchell, G. (2009). Investigation of Hamming, Reed-Solomon, and turbo forward error correcting codes. *Army Research Laboratory, ARL-TR-4901*.

Mohammad, A. S., Reddy, N., James, F., & Beard, C. (2018). Demodulation of faded wireless signals using deep convolutional neural networks. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018*, 969–975. https://doi.org/10.1109/CCWC.2018.8301731

Nachmani, E., Be'Ery, Y., & Burshtein, D. (2016). Learning to decode linear codes using deep learning. *54th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2016*, 341–346. https://doi.org/10.1109/ALLERTON.2016.7852251

Nachmani, E., Marciano, E., Lugosch, L., Gross, W. J., Burshtein, D., & Be'Ery, Y. (2018). Deep learning methods for improved decoding of linear codes. *IEEE Journal on Selected Topics in Signal Processing*, *12*(1), 119–131. https://doi.org/10.1109/JSTSP.2017.2788405

Nassar, C. (2001). Chapter 5-getting it from here to there: Modulators and demodulators. *Telecommunications demystified* (pp. 115–170). https://doi.org/10.1016/B978-0-08-051867-1.50011-1

National Instruments. (2019). *How AI is Starting to Influence Wireless Communications*. IEEE Spectrum. https://spectrum.ieee.org/computing/software/how-ai-is-starting-to-influence-wireless-communications

O'Shea, T., & Hoydis, J. (2017). An introduction to deep learning for astronomy. *IEEE Transactions on Cognitive Communications and Networking*, *3*(4), 563–575. https://doi.org/10.1109/TCCN.2017.2758370

O'Shea, T. J., Erpek, T., & Clancy, T. C. (2017). Deep learning based MIMO communications. *ArXiv, abs/1707.07980.*

Parker, M. (2017). Error-correction coding. *Digital signal processing 101* (2nd ed., pp. 129–147). Newnes. https://doi.org/10.1016/B978-0-12-811453-7.00012-3

Plonus, M. (2001). Chapter 9 digital systems. *Electronics and communications for scientists and engineers* (pp. 327–403). Academic Press. https://doi.org/10.1016/B978-012533084-8/50020-3

Proakis, J. G., & Salehi, M. (2001). *Communication Systems Engineering*. Prentice-Hall.

Qin, Z., Ye, H., Li, G. Y., & Juang, B. H. F. (2019). Deep learning in physical layer communications. *IEEE Wireless Communications*, *26*(2), 93–99. https://doi.org/10.1109/MWC.2019.1800601

Ramachandran, P., Zoph, B., & Le, Q. V. (2018). Searching for activation functions. *6th International Conference on Learning Representations, ICLR 2018*. https://arxiv.org/pdf/1710.05941.pdf

Ramsundar, B., & Zadeh, R. B. (2018). Chapter 4. fully connected deep network. *TensorFlow for deep learning*. O'Reilly Media, Inc. https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information (Switzerland)*, *11*(4), 1–48. https://doi.org/10.3390/info11040193

Sarkar, K. (2018). *ReLU : Not a differentiable function: Why used in gradient based optimization? and other generalizations of ReLU.* Medium. https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec

Sattiraju, R., Weinand, A., & Schotten, H. D. (2019). AI-assisted PHY technologies for 6G and beyond wireless networks. *The 1st 6G WIRELESS SUMMIT*.

Sun, J., Zhang, Y., Xue, J., & Xu, Z. (2020). Learning to search for MIMO detection. *IEEE Transactions on Wireless Communications*, *19*(11), 7571–7584. https://doi.org/10.1109/TWC.2020.3012785

Sundareshan, B. (1992). Digital modulation - baseband techniques. *IETE Journal of Education*,

33(1), 35–44. https://doi.org/10.1080/09747338.1992.11436355

Tse, D., & Pramod, V. (2005). *Fundamentals of wireless communication* (Vol. 9780521845). Cambridge University Press. https://doi.org/10.1017/CBO9780511807213

Vaz, A. C., Nayak, C. G., & Nayak, D. (2019). Hamming code performance evaluation using artificial neural network decoder. *2019 15th International Conference on Engineering of Modern Electric Systems, EMES 2019*, 37–40. https://doi.org/10.1109/EMES.2019.8795093

Wang, C. X., Renzo, M. Di, Stańczak, S., Wang, S., & Larsson, E. G. (2020). Artificial intelligence enabled wireless networking for 5G and beyond: Recent advances and future challenges. *IEEE Wireless Communications*, *27*(1), 16–23. https://doi.org/10.1109/MWC.001.1900292

Wang, H., Wu, Z., Ma, S., Lu, S., Zhang, H., DIng, G., & Li, S. (2019). Deep learning for signal demodulation in physical layer wireless communications: Prototype platform, open dataset, and analytics. *IEEE Access*, *7*, 30792–30801. https://doi.org/10.1109/ACCESS.2019.2903130

Wang, T., Wen, C. K., Wang, H., Gao, F., Jiang, T., & Jin, S. (2017). Deep learning for wireless physical layer: Opportunities and challenges. *China Communications*, *14*(11), 92–111. https://doi.org/10.1109/CC.2017.8233654

Westall, J. (2010). An introduction to Galois fields and Reed-Solomon coding. *School of Computing, Clemson University Clemson, SC*, 1–16.

Witten, H. I., Frank, E., Hall, M. A., & Pal, C. J. (2017). Chapter 10 deep learning. *Data mining* (4nd ed., pp. 417–466). https://doi.org/10.1016/B978-0-12-804291-5.00010-6

Woods, J. W. (2012). Chapter 13 video transmission over networks. *Multidimensional signal, image, and video processing and coding* (2nd ed., pp. 529–573). https://doi.org/10.1016/B978-0-12-381420-3.00013-8

Xu, W., Zhong, Z., Be'ery, Y., You, X., & Zhang, C. (2018). Joint neural network equalizer and decoder. *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*. https://doi.org/10.1109/ISWCS.2018.8491056

Ye, H., & Li, G. Y. (2017). Initial results on deep learning for joint channel equalization and decoding. *IEEE Vehicular Technology Conference*. https://doi.org/10.1109/VTCFall.2017.8288419

Ye, H., Li, G. Y., & Juang, B. H. (2018). Power of deep learning for channel estimation and signal detection in OFDM systems. *IEEE Wireless Communications Letters*, *7*(1), 114–117. https://doi.org/10.1109/LWC.2017.2757490

Zhang, M. L., & Zhou, Z. H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, *18*(10), 1338–1351. https://doi.org/10.1109/TKDE.2006.162

Zhang, M. L., & Zhou, Z. H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837. https://doi.org/10.1109/TKDE.2013.39

Zhang, Y., Doshi, A., Liston, R., Tan, W.-T., Zhu, X., Andrews, J. G., & Heath, R. W. (2020). DeepWiPHY: Deep learning-based receiver design and dataset for IEEE 802.11ax systems. *IEEE Transactions on Wireless Communications*, *20*(3), 1596–1611. https://doi.org/10.1109/TWC.2020.3034610

Zhu, X., Wang, T., Bao, Y., Hu, F., & Li, S. (2019). Signal detection in generalized gaussian distribution noise with Nakagami fading channel. *IEEE Access*, *7*, 23120–23126. https://doi.org/10.1109/ACCESS.2019.2895627

APPENDIX A: APPROVAL LETTER

MARSHALL UNIVERSITY
www.marshall.edu

Office of Research Integrity

July 26, 2021

Wenjie Xu
1517 6th Ave., Apt 2
Huntington, WV 25701

Dear Wenjie:

This letter is in response to the submitted thesis abstract entitled *"Artificial Intelligence Aided Receiver Design for Wireless Communication Systems."* After assessing the abstract, it has been deemed not to be human subject research and therefore exempt from oversight of the Marshall University Institutional Review Board (IRB). The Code of Federal Regulations (45CFR46) has set forth the criteria utilized in making this determination. Since the information in this study does not involve human subjects as defined in the above referenced instruction, it is not considered human subject research. If there are any changes to the abstract you provided then you would need to resubmit that information to the Office of Research Integrity for review and a determination.

I appreciate your willingness to submit the abstract for determination. Please feel free to contact the Office of Research Integrity if you have any questions regarding future protocols that may require IRB review.

Sincerely,

Bruce F. Day, ThD, CIP
Director

APPENDIX B: ACRONYMS

5G	Fifth Generation
IoT	Internet of Things
IoV	Internet of Vehicles
M2M	Machine to Machine
D2D	Device to Device
PHY	Physical Layer
AI	Artificial Intelligence
SISO	Single-input Single-output
SIMO	Single-input Multiple-output
MISO	Multiple-input Single-output
MIMO	Multiple-input Multiple-output
ML	Machine Learning
DL	Deep Learning
DNN	Deep Neural Network
SNR	Signal-to-noise Ratio
MLC	Multi-label Classification
MOC	Multi-output Classification
BER	Bit Error Rate
DeModCoder	Joint Demodulator and Channel Decoder
DeTecModCoder	Joint Detector (Equalizer), Demodulator, and Channel Decoder
AWGN	Additive White Gaussian Noise
GGN	Generalized Gaussian Noise

RS code	Reed-Solomon Code
PSK	Phase Shift Keying
BPSK	Binary Phase Shift Keying
QPSK	Quadrature Phase Shift Keying
QAM	Quadrature Amplitude Modulation
CSI	Channel State Information
ReLU	Rectified Linear Unit
PReLU	Parametric Rectified Linear Unit
NR	New Radio
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
BICM	Bit Interleaved Coded Modulation
LDPC	Low Density Parity Check
DCO-OFDM	Direct Current-Biased Optical Orthogonal Frequency Division Multiplexing
LLR	Likelihood Ratio
MMSE	Minimum Mean Squared Error
Rx2	2 Received Antennas
Rx4	4 Received Antennas

APPENDIX C: TOOLS AND SERVICE CONFIGURATIONS

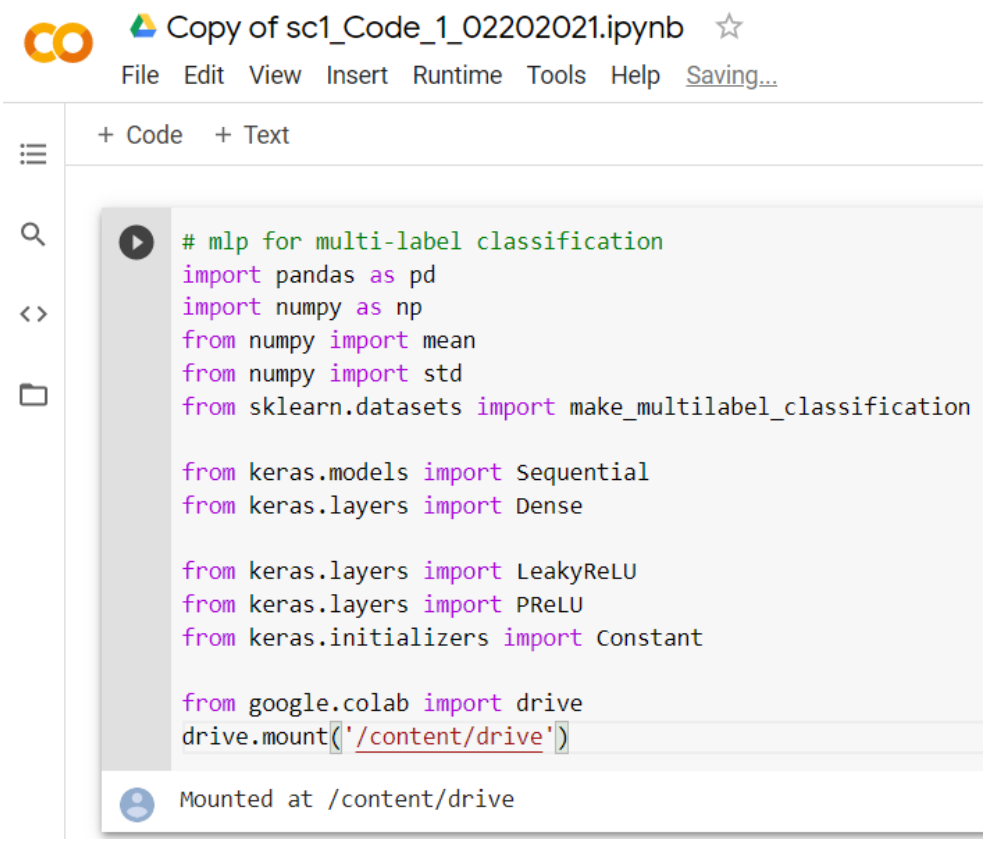

Figure 18. Tools and Cloud Services Using to Train NN.

As presented in Figure 18, all of our user-cases were designed based on Google Collaboratory (Colab) (Chamier et al., 2020), where insert datasets store in Google Drive (Harkous & Aberer, 2017). We applied Pandas and NumPy libraries in the Python framework with Tensorflow/Keras modules to implement simulations in our research (Abadi et al., 2016; Raschka et al., 2020). Furthermore, the free computing resources that we applied in Google Colab contains a 2.2 GHz Intel Xeon dual-core central processing unit (CPU) with 12.69 GB available random-access memory (RAM) and 107.72 GB Colab disk storage, and those configurations were gathered using methods written from Bakarola, 2021.

APPENDIX D: PARAMETERS OF SYSTEM MODEL

```
# get the model
def get_model(n_inputs, n_outputs):
    model = Sequential()
    model.add(Dense(200, input_dim=n_inputs, kernel_initializer='he_uniform'))
    model.add(PReLU(alpha_initializer=Constant(value=0.25)))
    model.add(Dense(200, input_dim=200, kernel_initializer='he_uniform'))
    model.add(PReLU(alpha_initializer=Constant(value=0.25)))
    model.add(Dense(n_outputs, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam')
    return model
```

```
[ ]  def bigTraining(X_train,y_train):
        n_inputs, n_outputs = X_train.shape[1], y_train.shape[1]
        model = get_model(n_inputs, n_outputs)
        model.fit(X_train, y_train, verbose=0, epochs=500, batch_size=64)
        return model
```

Figure 19. Parameters of System Model.

Figure 19 presents a sample model for the scenario of Reed-Solomon Code and QPSK at Rx2. Here, we first defined a system model, where arranged 2 hidden layers and 200 neurons at each layer applying the PReLU activation function. When the training process started, the model was trained in a supervised learning manner and using a backpropagation algorithm with 500 epochs and 64 batch sizes for each epoch.

APPENDIX E: SUPPLEMENTARY PROCESS OF TRAINING

```
for idx,val in enumerate(SNRdB):
    dataFrame_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Task3/
    dataFrame_target = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Task
    train_data = dataFrame_data.values
    train_target = dataFrame_target.values

    train_data_big = np.concatenate((train_data_big,train_data),axis=1)
    train_target_big = np.concatenate((train_target_big,train_target),axis=1)

train_data_big_V2 = train_data_big[:,data_size:(12*data_size)]
train_target_big_V2 = train_target_big[:,data_size:(12*data_size)]

X = train_data_big_V2.transpose()
y = train_target_big_V2.transpose()
model = bigTraining(X,y)
```

```
[ ] for idx,val in enumerate(SNRdB):
        dataFrame_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Task3/
        dataFrame_target = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Task
        train_data = dataFrame_data.values
        train_target = dataFrame_target.values

        X = train_data.transpose()
        y = train_target.transpose()
        yhat = model.predict(X)
        yhat = yhat.round()
        err = (y != yhat).sum()
        ber[idx] = err/np.size(y)
        ber_csv.append(ber[idx])
```

Figure 20. Sample Process of Training a Single NN.

A sample process of training a single NN is presented in Figure 20. First, for each SNR over a certain range of SNR values, we gathered received data symbols and transmitted data bit sequences from CSV file which stored in Google Drive (Harkous & Aberer, 2017), and those two types of data symbols were fed into big input and output training sets, respectively. After the system model was trained, another combination of received symbols and transmitted data bits were fed into the trained model to test the performance of our proposed design.

APPENDIX F: SUPPLEMENTARY DATA INFORMATION

- All the codes and data can be found in the following link from Google Drive, containing codes of simulation for every user-case, input and output datasets for different types of noise, and the simulation results: https://drive.google.com/drive/folders/1JLj5OeHw2cvIJFF5s1aPlxsBDZ_spWzm?usp=sharing
- Note that the results we obtained are not the termination of our work. For some considered cases, such as Reed-Solomon Code with QPSK at Rx4 case, we have not acquired a good performance yet. One of the potential reasons is my limited computing resources of simulation, for instance, the maximum runtime of Google Colab is 12 hours. If they were trained for a longer amount of time, such as one day or two days, the results may show improved performance.