University of BRISTOL

End-to-end Learning with Differentiable OFDM Systems

Zihao Liu

May 2023

Final year project thesis submitted in support of the degree of

Bachelor of Engineering in Electrical and Electronic Engineering

Department of Electrical & Electronic Engineering

University of Bristol

DECLARATION AND DISCLAIMER

DECLARATION AND DISCLAIMER

Unless otherwise acknowledged, the content of this thesis is the original work of the author. None of the work in this thesis has been submitted by the author in support of an application for another degree or qualification at this or any other university or institute of learning.

The views in this document are those of the author and do not in any way represent those of the University.

The author confirms that the printed copy and electronic version of this thesis are identical.

Signed: Zihao Liu

Dated: May 5th 2023

ACKNOWLEDGEMENTS

First and foremost, I want to express my sincerest gratitude to my supervisor, Professor Robert Piechocki. His support in providing valuable reading materials and insightful advice has been instrumental in optimizing my thesis. He also suggested some valuable websites to study machine learning as well as recommended using Google Colab as an alternative source of hardware when my computer was unable to keep up.

I also extend my gratitude to Dr Waqas Bin Abbas for taking the time to interview me for my poster assessment. His kind offer for further discussions on this project was greatly appreciated. He shared some incredibly useful ideas for optimizing my model that facilitated my work.

Last but not least, I want to thank Dr Xenofon Vasilakos for his helpful feedback on my poster assessment. His recommendation to use confidence intervals for evaluating the loss during model training was very insightful and undoubtedly enhanced the quality of my work. I'm grateful for his time and expertise.

ABSTRACT

From natural language processing to image classification and recognition, deep learning (DL) has proven its ability to achieve excellent performance in many real-life applications. In recent years, DL attracted many researchers in wireless communication to study the feasibility of integrating DL methodologies in the optimisation of wireless systems. Compared to the traditional modular approach of optimising the wireless system, DL provided simplicity in optimising the wireless system in an end-to-end manner by combining the transmitter, channel, and receiver into a trainable autoencoder. Although many researchers have already obtained promising results from using autoencoders in different aspects of communication systems, few of them addressed the performance of autoencoders in advanced channel models using high-order modulation schemes. Therefore, this research seeks to evaluate the performance of using a Residual Convolutional Neural Network (ResNet) based receiver in the autoencoder that is simulated in advanced Orthogonal Frequency Division Multiplexing (OFDM) systems at high-order Quadrature Amplitude Modulation (QAM). This research begins with conducting experiments with different configurations of the ResNet using a simple Additive White Gaussian Noise (AWGN) channel model to gain insights on tuning the hyper-parameters. The conclusions drawn from the experiments are then applied to advanced OFDM channel models that simulate realistic channel environments with 1024-QAM. The trained model is benchmarked against two conventional baselines using a Bit Error Rate (BER). The results demonstrated that the neural receiver outperforms the traditional method in high-order QAM, where the symbols are more prone to noise. Compared to channel coding and channel estimation methods, it is shown that the neural receiver is able to achieve better BER at lower energy per bit to noise power spectral density (Eb/No) ratio .

CONTENTS

LIST OF ACRONYMS

3GPP	Third Generation Partnership Project
API	Application Programming Interface
AWGN	Additive White Gaussian Noise
BCE	Binary Cross Entropy
BER	Bit Error Rate
BPSK	Binary Phase-Shift Keying
BS	Base Station / Batch Size
CDL	Cluster Delay Line
CLR	Cyclical Learning Rate
CNN	Convolutional Neural Network
CSI	Channel State Information
DL	Deep Learning
Eb/No	Ratio of Energy per Bit to Noise Power Spectral Density
FEC	Forward Error Correction
FNN	Feedforward Neural Network
LDPC	Low-Density Parity Check
LLR	Log-likelihood Ratio
LMMSE	Linear Minimum Mean Square Error
LOS	Line of Sight
LR	Learning Rate
LS	Least Squares
LSTM	Long Short-Term Memory Neural Network
MIMO	Multiple-Input-Multiple-Output
NLOS	Non-line of Sight
OFDM	Orthogonal Frequency Division Multiplexing
QAM	Quadrature Amplitude Modulation
ReLU	Rectified Linear Unit
ResNet	Residual Convolutional Neural Network
RMa	Rural Macrocell

RNN	Recurrent Neural Network
RS	Residual Blocks
SGD	Stochastic Gradient Descent
SIMO	Single-Input-Multiple-Output
UMa	Urban Macrocell
UMi	Urban Microcell
UT	User Terminal

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 OVERVIEW

In wireless communications, high data rates are now achievable through modulation techniques based on Orthogonal Frequency Division Multiplexing (OFDM) [1] and high order Quadrature Amplitude Modulation (QAM) [2]. However, high-order modulation schemes are more prone to noise, where symbols close in Euclidean distance can be misinterpreted at the receiver, making it difficult to ensure the quality and reliability of transmitted data.

This is especially important in urban areas. The high number of users and demand for data rates often means that signals can take multiple paths as they pass through buildings and other structures, causing multi-path signals that introduce distortion, delays, and non-linearities that degrade the quality and reliability of received data symbols. Although modern channel coding and estimation methods such as Low-Density Parity Check (LPDC), Least Square (LS) estimation and Linear Minimum Mean Square Error (LMMSE) can effectively estimate the received signal based on linear combination of the transmitted signal and the noise, the presence of non-linearities and complexities in the environment can still lead to inaccurate estimation if only linear-based methods are used. Therefore, it is imperative to search for better solutions to overcome these challenges.

In recent years, deep learning (DL) has proven to be an effective method for solving classification, prediction and clustering problems with many non-linearities. Therefore, DL is considered a feasible solution to communication systems and encouraged many researches in this domain. Apart from DL's promising potential, many researchers are triggered to apply DL techniques to optimise wireless communication systems due to its simplicity compared to the traditional approach. With DL, the optimisation process is done in an end-to-end manner where the entire system is trained as a single entity known as an autoencoder rather than individual blocks. Furthermore, unlike traditional modular optimisation, the autoencoder can be trained on any realistic channel model thanks to powerful wireless communication simulation tools Sionna [3] and open-source DL libraries Keras [4] and Tensorflow [5].

The first study to explore the use of autoencoders in communication systems for the physical layer [6] involved training a conventional neural network with several dense layers on a simple AWGN channel model. The success in the implementation lead to papers discussing various aspects of machine learning and its potential benefits in wireless communications [27], where the author talked about supervised learning, unsupervised learning and reinforcement learning and how they can be applied to communication systems. This inspired further research to explore the application of autoencoders in more complex set-ups, such as OFDM systems [29][30], Multiple-Input-Multiple-Output (MIMO) systems with space division multiplexing [7], and multi-user scenarios [8]. Many subsequent studies have since been conducted using different forms of neural networks, such as Convolutional Neural Network (CNN) [9][31] and Long-Short Term Memory (LSTM) neural network [10][22], in order to adapt to the diverse characteristics of the different communication systems.

Although autoencoders have been studied for use in many communication systems, few have investigated their effectiveness at high order modulation schemes under advanced channel models. Some researchers explored the use of probabilistic models and autoencoders for modulations up to 1024-QAM [11], while others proposed a residual CNN to address symbol denoising up to 256-QAM [40], but these were only tested on a simple AWGN channel model. Another study analysed the performance of using DL with high-order QAM of up to 64-QAM in massive MIMO system [41] that maximizes spectral efficiency, but again only simulated with a Rayleigh fading channel where no spatial characteristics of multipath propagation were modelled. Therefore, this study aims to extend previous researches and evaluate the performance of high-order QAM under advanced urban environment channel models defined by the Third Generation Partnership Project (3GPP)

1.2 AIMS AND OBJECTIVES

This study aims to extend and improve an end-to-end software-based OFDM communication system that operates at high-order modulation schemes, based on the model presented in [12]. The primary objective is to compare the performance of a neural receiver with traditional channel coding and estimation methods, using realistic channel models such as Cluster Delay Line (CDL) and Urban Microcell (UMi) that account for spatial correlation, multipath propagation, scattering, reflection, and Non-line of Sight (NLOS) propagation. The study seeks to highlight the benefits of the proposed system in high-order modulation schemes under realistic channel conditions.

1.3 THESIS OUTLINE

The rest of the thesis is structured as follows:

- ✧ Chapter 2 explains the architecture of the ResNet neural receiver and discusses the underlying principles of neural network operation and its hyperparameters.
- ✧ Chapter 3 describes the architecture of the autoencoder and presents the channel models and benchmarks used for evaluating the performance of the neural receiver.
- ✧ Chapter 4 conducts experiments on tuning the hyperparameters of the neural network with a simple AWGN model at 256-QAM to find the best configuration of the neural network based on the theories established in Chapter 2.
- ✧ Chapter 5 implements the findings obtained from Chapter 4 into advanced channel models, such as CDL and UMi with 1024-QAM, and presents the results of simulations with discussions.
- ✧ Chapter 6 presents several additional works that has been done to gain more understanding of the neural receiver. This relates to attempts made on using the neural receiver with lower order QAM and MIMO systems.
- ✧ Chapter 7 concludes the thesis and provides suggestions for future work that can be carried out in this area.

2 BACKGROUNDS ON THE NEURAL RECEIVER

2.1 OVERVIEW OF DEEP LEARNING

DL is a method that combines statistics, computing and algorithms theories to enable a particular model to learn iteratively and find the hidden insights and underlying patterns in datasets. Inspired by the structure of the human brain, DL uses layers of neurons that are interconnected with each other forming a network, the first and last layer of the network is the input and output layers, and the layer in between the outermost layer is known as the hidden layer. A neural network is commonly regarded as deep when a high number of hidden layers is used. FIGURE 2.1 depicts the architecture of a neural network with six input neurons, one output neuron, and two hidden layers.

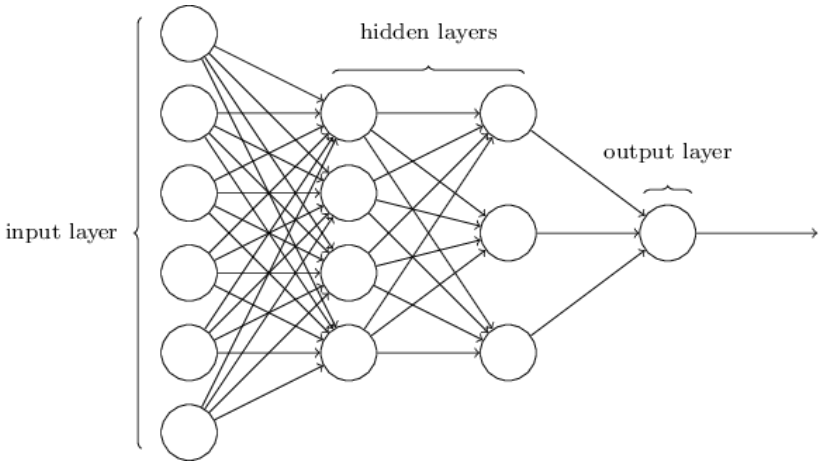


FIGURE 2.1: *Prototype of a neural network [13] (Ch. 1)*

In a neural network designed for wireless communication, the input layer usually takes in raw binary bits in the form of a high-dimensional matrix. The shape of the matrix depends on the modulation scheme used, the number of transmitter and receiver antennas, and the number of users. Each neuron in the network has a unique set of weights and biases used to compute a value for each input. This value is then compared to a certain threshold using activation functions, determining whether the value should be propagated further through the network. The number of neurons in the hidden layer is one of the neural network's hyper-parameters. It can be adjusted according to different scenarios in the channel models. A larger number of neurons can help capture more complex patterns in the data but may also result in overfitting. In comparison, fewer neurons may reduce the computational complexity but may

result in underfitting. In addition, the number of neurons in the output layer equals the number of bits in the modulation scheme. For instance, if 64-QAM is used, the output layer will have six neurons. Once all the values reach the output layer of the network, a log-likelihood (LLR) ratio is used to make soft decisions on classifying the received symbols using a probability distribution function or confidence level. After this classification, the received symbols are compared to the transmitted symbols, and a loss function called binary cross entropy (BCE) is used to calculate the difference between them.

The goal of the neural network is to minimize the BCE loss function, which measures the difference between the received and transmitted signals. To do this, the Stochastic Gradient Descent (SGD) algorithm is used together with backpropagation to iteratively adjust the weights and biases of each layer in the network to minimize the loss function. In this way, the neural network can learn to make more accurate predictions over time.

In this chapter, Section 2.2 will study the underlying principle of SGD and backpropagation. Section 2.3 will delve into a hyper-parameter closely related to SGD called the learning rate. Section 2.4 will look at several commonly used types of neural networks and introduce the neural network used in this research, the ResNet. Finally, Section 2.5 will explore a source of inspiration based on the reviewed literature.

2.2 SGD AND BACKPROPAGATION

To understand SGD, it's worth first looking at gradient scent using a simple example. Let's assume there exists a loss function $L(w)$ that looks like a parabola, as shown in FIGURE 2.2, where w is the value of the weight in the neural network. The goal is to find the value of the weights that achieves the minimum loss on this curve. Using gradient descent, one can take steps proportional to the negative of the gradient of the function and move gradually towards the minimum point w_n iteratively.

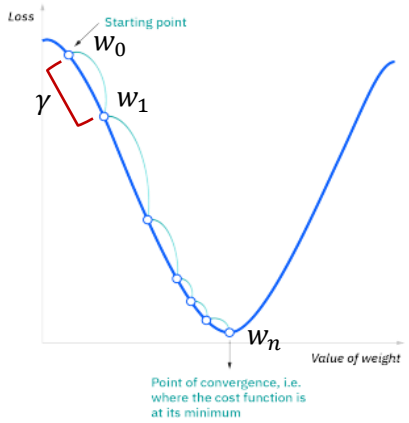


FIGURE 2.2: *Loss function with a shape of a parabola, edited based on [14]*

Usually, the minimum point w_n is unknown. So, to begin with the gradient descent, a random starting point w_0 is selected. Then, the gradient of the loss function at w_0 is computed by $\frac{\delta L}{\delta w_0}$ to determine the direction of the step. The size of the step is controlled by a parameter γ called learning rate. Therefore, in the first iteration, the algorithm moves from w_0 to w_1 by taking step size of $\gamma \frac{\delta L}{\delta w_0}$. The process is repeated until the algorithm converges to w_n.

SGD is similar to gradient descent in that it updates neural network parameters to minimize the loss function. However, while gradient descent computes the gradient using the entire training dataset, SGD computes gradients on a small, randomly selected subset. This essentially reduces the pressure on computational resources and allow more frequent updates of parameters.

A neural network comprises many interconnected neurons that work together to make predictions. When the output layer generates a prediction and calculates a loss, the parameters of the entire neural network need to be updated. However, it is not possible to update all parameters simultaneously. To address this issue, a technique called backpropagation is used. Backpropagation is an algorithm that uses SGD to update the weights and biases of neurons layer by layer, starting from the last layer. For example, the gradient of the penultimate layer depends on the gradient of the final layer, and since the gradient of the final layer has already been calculated, it can be used for the penultimate layer by applying the chain rule of differentiation. This process is repeated until the first layer has been updated. As a result, backpropagation and SGD is jointly used to help the neural network learn from the dataset by minimising the loss function repeatedly through many iterations.

2.3 THE LEARNING RATES

Section 2.2 briefly mentioned the learning rate, and this section will provide a more detailed discussion of this hyper-parameter. In general, the learning rate needs to be appropriately tuned before the training process to achieve optimal results because it determines the step size of each update during the gradient descent calculation.

A small learning rate, shown on the right in FIGURE 2.3, can always ensure that the model converges to the minimum point, but with the cost of a slow and long convergence process due to the need for more iterations. Additionally, a small learning rate may cause the model to get stuck in a local minimum point, limiting its ability to escape and find a better or global minimum point of a loss function.

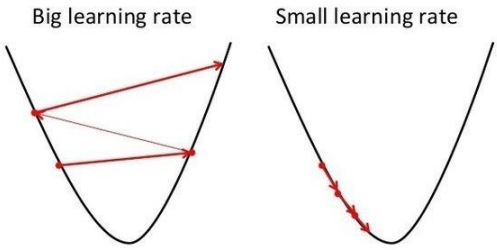


FIGURE 2.3: *Two plots demonstrating the behaviour of big and small learning rate [15]*

On the other hand, a high learning rate can reduce the time needed to converge, but if it's too high, it may cause divergence problems, where the model cannot converge to a minimum point, this example is demonstrated in the left plot in FIGURE 2.3. There is no one-size-fits-all learning rate, and for most applications, extensive trial and error is necessary to find the most suitable value. This process can be time-consuming because deep neural network training usually requires a large number of iterations which can take hours or even days to train.

In [16], researchers aimed to address the issue of determining an appropriate learning rate for neural network training. They compared the accuracy of popular learning rate scheduler using CNN and trained each scheduler on the MNIST and CIFAR-10 datasets for classification problems.

Out of the six candidates tested, the cyclical learning rate (CLR) scheduler proved to be the most effective in terms of both accuracy and convenience. With CLR, there is no need for manual tuning, as the value cyclically varies between a reasonable range as shown in FIGURE 2.4. The boundaries for this range can be found easily within a few iterations using the "learning rate (LR) range test" [17] without having to train the entire model. Furthermore, the CLR scheduler was shown to be able to reduce the number of epochs required for training while still maintaining high accuracy.

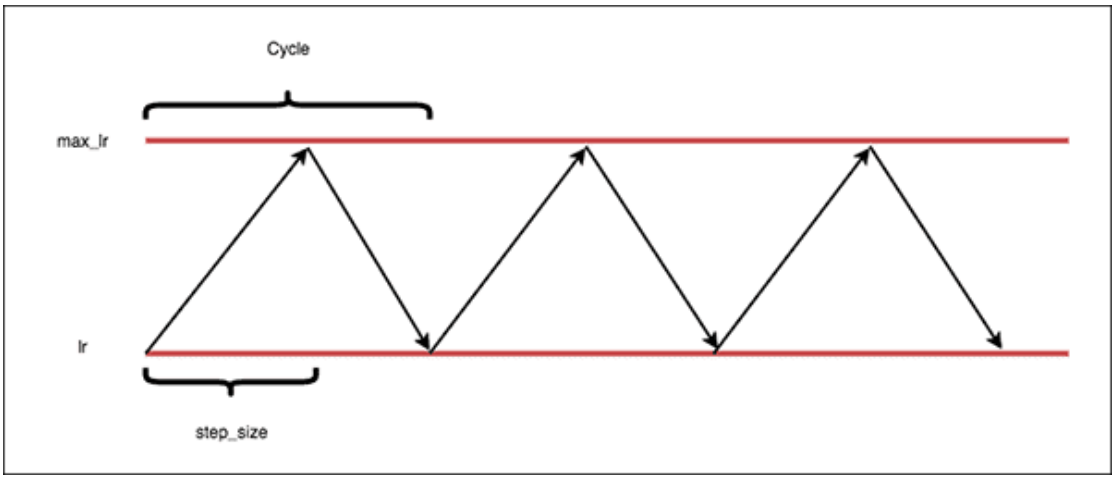
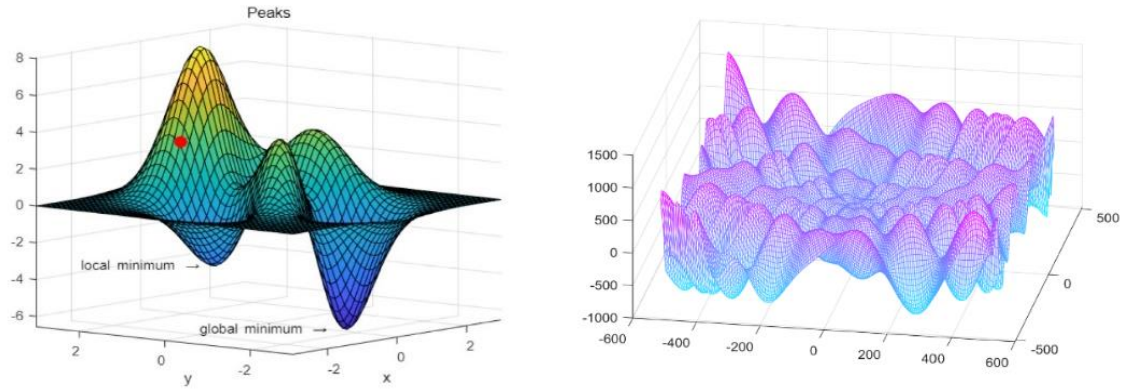


FIGURE 2.4: *CLR with triangular policy [18]*

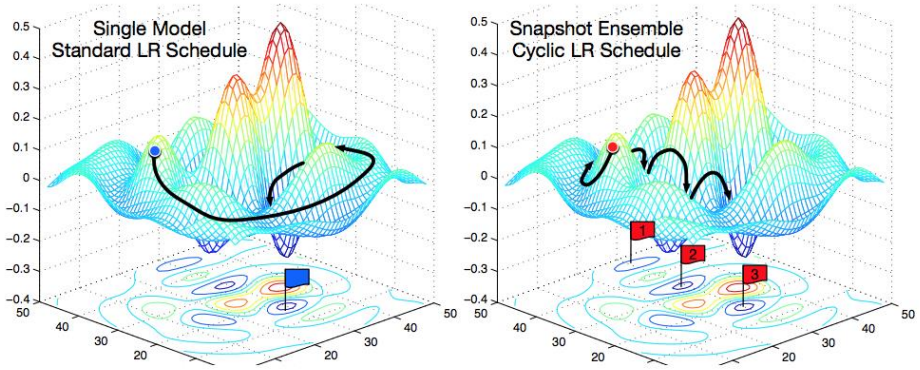
Unlike the parabola described in the Section 2.2, the loss functions in DL can be extremely complex, with multiple minimum points. The global minimum, the point with the lowest minimum, is what the algorithm should converge to. However, with a small learning rate, the algorithm may converge to a local minimum and mistakenly assume it has found the global minimum. This is demonstrated in FIGURE 2.5*a*, which shows a 3D plot of a peak function implemented in MATLAB. Assuming that the algorithm randomly chooses the red dot as the starting point, if the learning rate is too small, the algorithm is likely get stuck in the local minimum labelled on the graph and never escape.



(a) *(b)*

FIGURE 2.5: *Two 3D plots demonstrating complex loss functions with multiple minimum points (a) peak function implemented in MATLAB (b) the Eggholder function (both plots are self-simulated)*

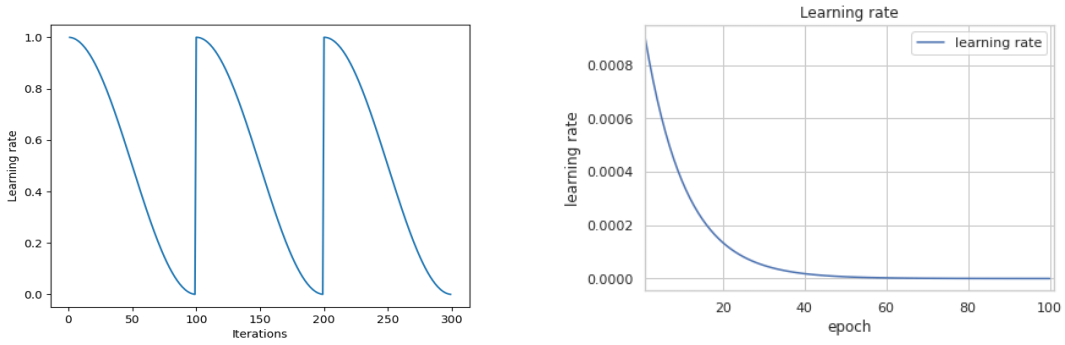
This is when learning rate schedulers like CLR comes in handy. These methods allow the learning rate to vary cyclically with time, creating more opportunities for the algorithm to escape local minimum and look for better solutions. In FIGURE 2.5*b*, the egg-holder function, commonly used to test the effectiveness of global minimization algorithms, is shown as another example of a loss function with multiple minimum points. These learning rate schedules can be helpful in finding the global minimum for such complex loss functions.



(a) *(b)*

FIGURE 2.6: *Comparison of the path of convergence of SGD for (a) constant learning rate (b) cyclical learning rate [19]*

Keras offers several Application Programming Interface (API) options for learning rate schedulers that can be conveniently utilized with an Adam optimizer [33]. However, Keras does not provide an API for the CLR scheduler. Therefore, this study will use two other common scheduler that is available in Keras: learning rate with warm restarts, and exponential decaying learning rate. The variation of learning rate with time for the two schedulers is described in FIGURE 2.7.



(a) *(b)*

FIGURE 2.7: *The variation of learning rate over time for (a) learning rate with warm restarts [20] (b) exponential decaying learning rate [21]*

2.4 DIFFERENT FORMS OF NEURAL NETWORKS AND THE RESNET

The neural network described in FIGURE 2.1 is a basic Feedforward Neural Network (FNN) that processes information in one direction, from input to output, through one or more hidden layers. FNNs work well for wireless communication when the dataset's dimension is small, such as in AWGN channels [28]. However, for high-dimensional input data, like in OFDM systems where the bandwidth is split into multiple subcarriers, the input matrix's dimension significantly increases, making it challenging to process the data using FNNs. To overcome this challenge, CNN was developed and widely used to classify high-dimensional input data. CNN uses kernels that convolve on the input matrix, reducing its dimensionality and computational complexity while preserving important features. Nowadays, CNNs are widely used in image classification and recognition applications, as well as in communication system studies.

In addition to FNNs, other neural network architectures are explicitly designed to handle sequential data by incorporating memory mechanisms. For example, Recurrent Neural Network (RNN) uses feedback loops to remember past inputs and use them to influence current input and output. RNNs excel in areas such as speech recognition and natural language processing, where there are dependencies between the current and past inputs. However, traditional RNNs can struggle with long-term dependencies, leading to the development of LSTM networks, a variation of RNNs, that better address this issue by using specialized memory cells to forget or retain information selectively. Researchers in

[22] demonstrated the successful use of a bidirectional LSTM network to detect NLOS signals in visible light communication. This neural network was able to process data bidirectionally, from input to output and vice versa, allowing it to remember patterns over time. Additionally, the network incorporated "forget gates" to selectively forget meaningless data stored in memory. Compared to traditional statistical equalization methods, the bidirectional LSTM network demonstrated superior performance in detecting NLOS signals in visible light communication.

In the classification problem of communication systems, as the dataset's complexity and number of features grow, one can choose to add more layers to their CNN to capture more details, this encourages the network to learn and extract increasingly abstract and high-level input representations as the data pass through more layers. Although this can lead to improved accuracy and performance, but adding too many layers to the network may give rise to the vanishing gradient problem.

As described in Section 2.2, backpropagation uses the chain rule of differentiation to update the weights and biases of neurons in each layer. However, this process can lead to vanishing gradients of the loss function with respect to the weights as they propagate through an increasing number of layers in the network. As the number of layers increases, the partial derivative of the loss function may approach zero, which can hinder the network's ability to learn and may lead to slow or stalled learning. One approach to address the vanishing gradient problem is to use the Rectified Linear Unit (ReLU) activation function, where the output of a neuron is equal to the input if the input is positive and zero otherwise. ReLU neurons do not saturate, unlike other activation functions, such as sigmoid or hyperbolic tangent functions, which can compress output values into a narrow range. However, even ReLU may be ineffective in overcoming the vanishing gradient problem in very deep neural networks.

To solve the vanishing gradient problem, residual or skip connections can be added to very deep neural networks. The concept of residual connections was first implemented in image classification applications [32]. Later, researchers in [23] proposed a ResNet model can be used in the context of communication systems. The working principle of the skip connections is illustrated in FIGURE 2.8. There are skip connections between every few stacked convolutional layers. These shortcut connections

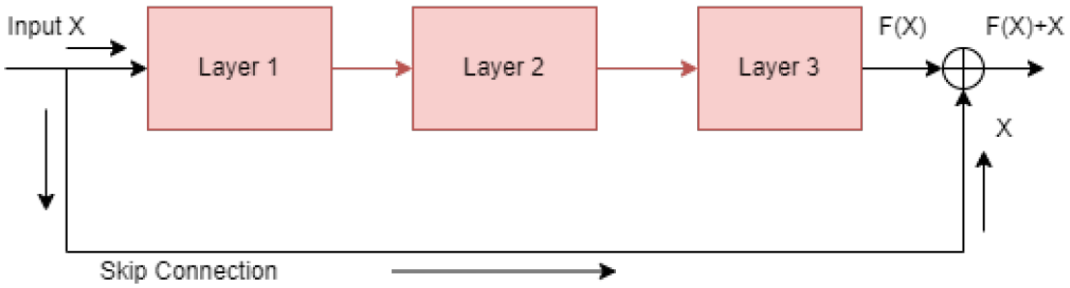


FIGURE 2.8: *A flowchart illustration of skip connections in ResNet [23]*

allow the input of a layer to be directly passed to a later layer, bypassing the intermediate layers. This helps to avoid the vanishing gradient problem by providing a direct path for the gradients to flow back through the network.

In this study, a ResNet structure extended from [24] will be used. The architecture of the ResNet is shown in FIGURE 2.9*a*. The ResNet consists of an input and output convolutional layer, with several residual blocks in between. The architecture of the residual block is illustrated in FIGURE 2.9*b*, where convolutional layers, normalization layers, and ReLU activation functions are employed. Also, every input to the residual block is directly added to its output to form shortcut connections. The combination of residual connections and ReLU activation function ensure that gradient will not vanish during backpropagation. In addition, the number of residual blocks can be adjusted based on the complexity of the channel by adding or removing them with a few lines of code. For example, for simple AWGN channels, fewer residual blocks can be used to avoid overfitting and reduce computational pressure, while for OFDM channels, more residual blocks can be used to capture more features and enhance the model's performance.

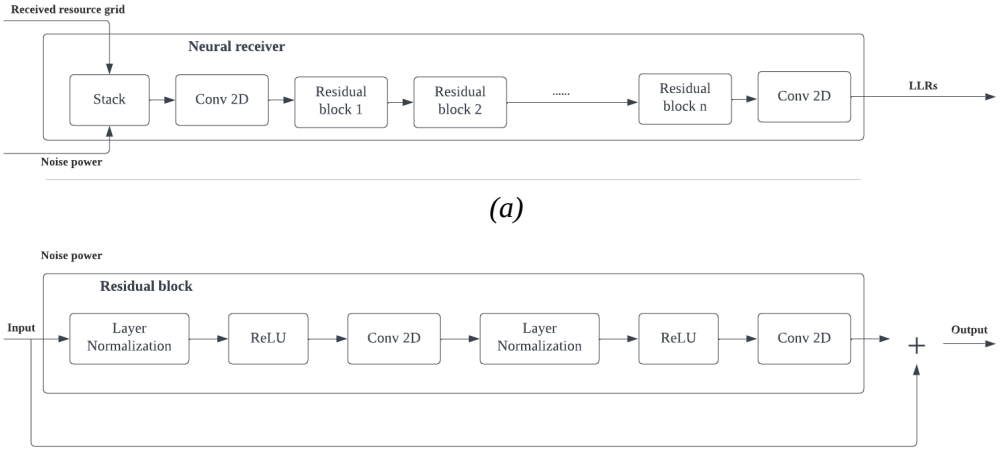


FIGURE 2.9: *Architecture of the ResNet implemented in this study. (a) shows the structure of the ResNet as a whole, (b) shows expands the residual block to show more details. Redrawn from [24]*

The ResNet architecture was chosen because advanced channel models and high-order modulation schemes are known to generate complex datasets with many features. ResNet is well suited to this type of data as it can effectively handle the vanishing gradient problem that arises when additional layers are added to the neural network to discover important features.

2.5 INSPIRATIONS

As discussed in Section 2.3, the CLR has been found to outperform other learning rate schedulers in terms of both accuracy and number of iterations [16]. While Keras does not provide an API for the CLR scheduler for its optimisers, this study attempted to design its own scheduler that implements the CLR. However, it was found that the optimizer of the model is typically defined outside the for loop that iterates over iterations, and in order to implement the self-designed CLR, the learning rate would need to be updated inside the for loop. Unfortunately, it was found that this Keras' model was not compatible with updating the learning rate inside the for loop.

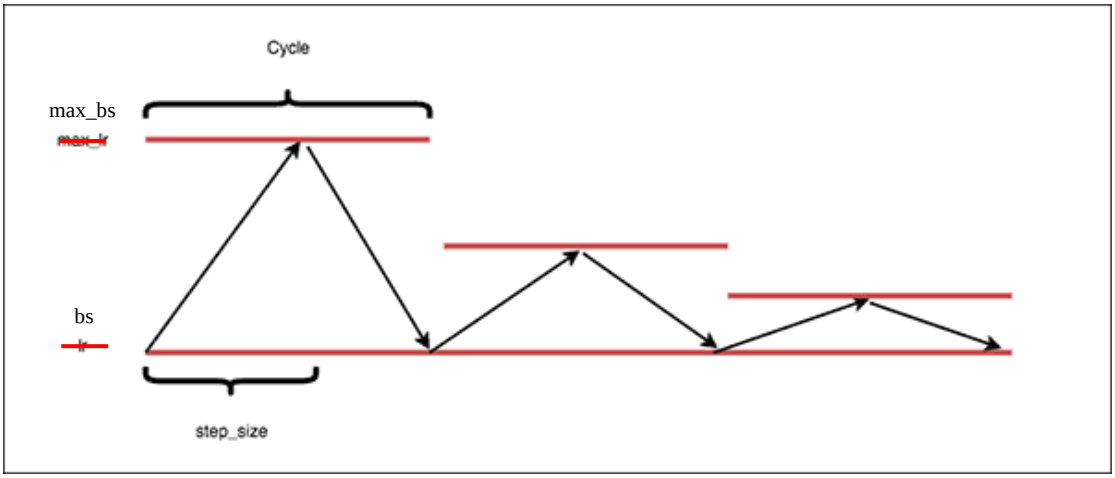


FIGURE 2.10: *Variation of cyclical decaying batch size with respect to time, edited based on [18]*

Although CLR cannot be used, this study was inspired to conducted additional experiments on the cyclical nature of other parameters in neural network. Through these experiments, it was discovered that Keras is fully compatible with adjusting the batch size within the for loop. This allows for changes in the number of samples fed into the neural network in parallel at once. In light of this, this study attempted to implement a cyclical pattern by varying the batch size over time with a cyclical decaying batch size. The resulting changes in the number of batch sizes over time are shown in FIGURE 2.10.

3 BACKGROUNDS ON THE END-TO-END SYSTEM

3.1 OVERVIEW

In recent years, open-source software has gained immense popularity, owing to the enthusiastic community that has led to the development of powerful tools. The autoencoder used in this thesis is based on Sionna [3], a TensorFlow-based library designed to simulate the physical layer of wireless communication systems. Sionna implements each building block in the communication system, such as binary source generator, encoder & decoder, OFDM resource grid, channel estimator, and symbol equalizers, as trainable Keras layers. The differentiable nature of these layers facilitates system optimization through DL. In addition, the use of SGD in Keras allows for the computation and minimization of loss functions at the receiving end, and updated gradients can be backpropagated to the entire end-to-end system to update the parameters of the trainable layers.

The rest of this chapter provides a comprehensive overview of the implementation of the end-to-end OFDM system using Sionna, and the parameters of each building block in the communication system. Section 3.2 presents the structure of the end-to-end system, and also discusses the two baselines used to bench mark the performance of the neural receiver. Section 3.3, Section 3.4, and Section 3.5 are dedicated to discussing the implementation of the transmitter, channel, and receiver, respectively.

3.2 STRUCTURE OF THE END-TO-END SYSTEM, AND THE BENCHMARKS

The structure of the end-to-end system is presented in FIGURE 3.1, where the autoencoder is comprised of a transmitter, a channel, and three types of receivers. Two of the receivers serve as baselines for benchmarking purposes, while the third receiver is ResNet-based and used for training.

Both baselines incorporate widely used Forward Error Correction (FEC), channel estimation and equalization techniques to ensure the reliability and quality of received signals. The only difference is that the first baseline is provided with the exact channel response, whereas the other baseline uses a technique called LS estimation to estimate the channel response.

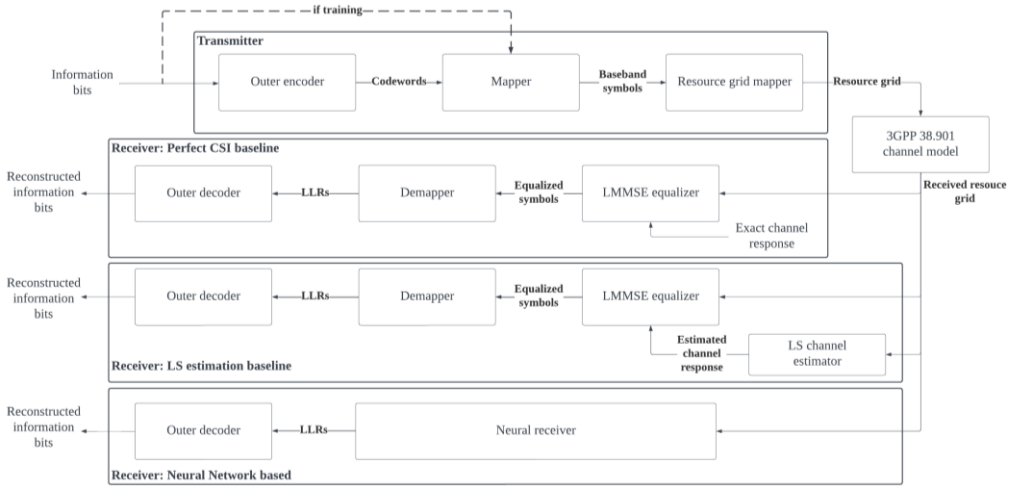


FIGURE 3.1: *Structure of the autoencoder used in this study, redrawn from [24]*

The exact channel response, also known as perfect Channel State Information (CSI), refers to a scenario where the receiver possesses a complete and accurate knowledge of the channel properties, including noise level, interference, phase shift, and delay. This is considered the ideal and most desirable situation that all FEC, channel estimation, and equalization techniques aim to achieve.

The following paragraphs discuss in more detail the working principle of FEC, channel estimation and equalization.

First, LDPC encoding and decoding are used for FEC. LDPC is an extension of Hamming's code, where parity bits are added to the end of the information bits to protect a set of information bits by counting the number of ones in that set. If the number of ones in the set is even, the parity check bit is set to one, and if it is odd, it is set to zero. This technique allows for the recovery of erased bits that have been corrupted while travelling through the noisy channel by solving a set of linear equations. The higher the number of parity bits used, the better the protection, but with the drawback of a lower code rate. This means that the number of meaningful information in a complete message block is low because most of them are parity bits. In Hamming's version, each set protects a vast number of information bits, which can lead to failures when there is more than one erasure in a set. Additionally, if one set covers many bits, the complexity of the linear equation that needs to be solved significantly increases, leading to unacceptable delays in recovering information bits. Therefore, LDPC code is introduced to address these issues by only covering a small range of information bits with each set, resulting in a low density of parity sets. This reduces the probability of multiple erasures in a parity set and lowers the computational complexity of solving sets of equations. In this way, the error correction can be much more efficient because the computer tends to perform better at solving a large number of simple operations compared to solving a small number of highly complex operations.

The second technique used in the baseline is the LS estimation. LS is a linear regression method that estimates the channel conditions by minimizing the sum of squares between the predicted signal and the actual transmitted signal. LS estimation works on fitting a best-fit line to the channel parameters, such as noise, delay, and interference, by assuming a linear relationship between the received and transmitted signals. By utilizing mathematical principles, LS estimation produces an estimate of the channel conditions, which is then passed on to the equalizer and decoder to produce predicted signals.

The final step in the baseline is implementing the LMMSE equalizer, which takes the estimated channel response from the LS estimator as input to process the channel response even further. Its primary objective is to minimize the error between the estimated signal and the actual transmitted signal using statistical principles based on Bayes' theorem. The LMMSE equalizer assumes that the transmitted symbols are a random process with a known statistical distribution. This equalization technique is particularly effective in mitigating inter-symbol interference caused by multipath propagation. Once the channel response is estimated, it is passed on to the LDPC decoding stage to perform hard decisions on the received signal.

3.3 TRANSMITTER

In Sionna, the information bits can be easily generated by selecting the desired modulation scheme. This study utilizes 256-QAM and 1024-QAM, which correspond to 8 and 10 bits per symbol, respectively.

For baselines, generated bits undergo LDPC encoding and are mapped onto a constellation map. In contrast, for the neural receiver, LDPC encoding is skipped and the uncoded bits are directly mapped onto the constellation map as shown by the dashed arrow in FIGURE 3.1. The resulting 256-QAM constellation map is shown in FIGURE 3.2. Each blue dot on the map represents a symbol comprising of eight bits. The small Euclidean distance between each symbol implies that even slight variations in the real and imaginary axes of the received signal can result in incorrect symbol interpretation. Consequently, high-order modulation schemes are more susceptible to interference than lower-order ones.

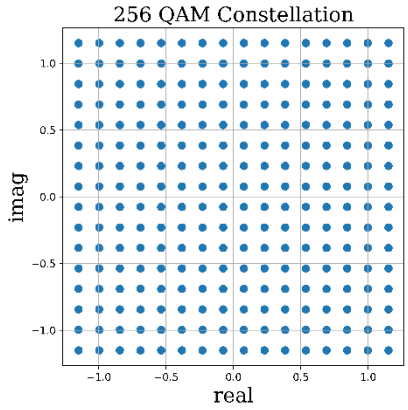


FIGURE 3.2: *Constellation plot of 256-QAM*

Then, the constellation in FIGURE 3.2 is mapped onto a two-dimensional OFDM resource grid matrix, which organizes how each subcarrier and symbol carries data or information. Unlike frequency division modulation, OFDM uses orthogonal frequency carriers, which allows the use of more subcarriers that are more closely spaced within a specific bandwidth. Each subcarrier carries an independent data stream, allowing multiple data streams being transmitted simultaneously, providing greater throughput in a single transmission. The OFDM resource grid is a two-dimensional matrix that specifies how subcarriers and symbols are allocated for modulation in the frequency and time domains, respectively. FIGURE 3.3 visualizes the OFDM resource grid, where the x-axis shows the symbol allocation and the y-axis shows the subcarrier allocation. In this study, 152 subcarriers and 14 OFDM symbols in used. From the 14 OFDM symbols, two will be used as pilot symbols. The reasoning behind these numbers is explained is detail in Section 5.2.

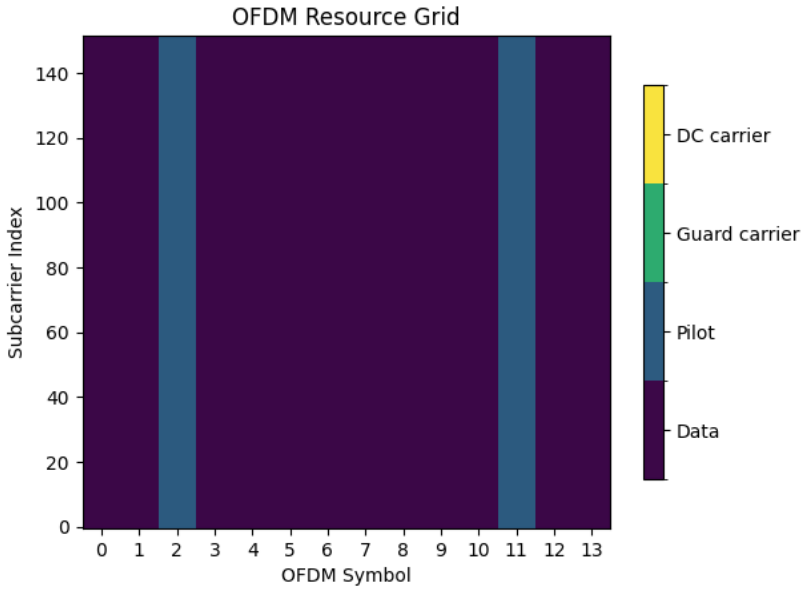


FIGURE 3.3: *OFDM resource grid with 152 subcarriers, 14 symbols, and pilots with indices (2,11)*

The resource grid also includes pilots, which are signals known by both transmitter and receiver, and it is transmitted along with the data signal. Pilots are used by the receiver to adjust the transmitted signal and compensate for channel variations, improving the quality of the received signal and reducing the effects of fading and interference. Pilots can also be used for synchronization, aligning the receiver's clock with the transmitters. In FIGURE 3.3, pilots are shown as blue lines, and this study uses symbols with indices two and eleven as pilots.

In [25], researchers examined how pilots can significantly reduce BER when using a neural receiver over multiple subcarriers. They also found that jointly optimizing the transmitter and the neural receiver can achieve reliable symbol detection without using pilots, considering an OFDM channel model that includes frequency selectivity and channel aging.

Once the OFDM resource grid is created, it is inserted into the software-based channel along with the self-defined noise level to simulate the phenomenon that may occur in realistic environments.

3.4 CHANNEL MODELS

The research will utilize the channel models specified in the 3GPP standard, which includes a variety of models that incorporate both line of sight (LOS) and non-line of sight (NLOS) scenarios. LOS scenarios refer to signal paths that do not encounter any obstructions as they travel from the transmitter to the receiver, resulting in minimal interference and signal loss. This type of scenario is typically observed in rural areas with open fields or low population density. Conversely, NLOS scenarios involve obstacles such as buildings, trees, and other obstructions that cause the signal to deviate, refract, and take multiple paths from the transmitter to the receiver. This scenario is often encountered in urban areas such as city centers or indoor offices.

The clustered delay line (CDL) is a channel model used in 3GPP, where the received signal consists of distinct delayed clusters. Each cluster is made up of several multipath components with identical delay but varying angle of departure and arrival. The CDL model comprises five delay profiles that simulate diverse characteristics of multipath propagation based on statistical analysis and measurements of radio propagation in different environments. Three of the profiles focus on NLOS scenarios, namely CDL-A, CDL-B, and CDL-C, while CDL-D and CDL-E are used to model LOS scenarios.

Moreover, there exist channel models that depict signal transmission behavior in cellular networks of different sizes. These models are known as Urban Microcell (UMi), Urban Macrocell (UMa), and Rural Macrocell (RMa). UMi models are intended for small, low-power cells that cover limited areas

in densely populated urban settings, such as street canyon, indoor office or city centers. UMa models, in contrast, encompass larger cells with higher power levels that cover more extensive areas, such as suburban regions. Lastly, RMa models represent wireless network environments in rural areas where the population density is low and the base stations are even more distant apart than those in UMa.

Due to the aim of assessing the neural receiver's performance in high user density and data rate demand scenarios with NLOS multipath propagation, the study will primarily focus on urban areas. For this purpose, the study will prioritize the UMi and CDL-A channel models, as these are most relevant to the study's objectives. Specifically, CDL-A will be selected among the three NLOS CDL channel models is because it has the highest normalized delay (according to 3GPP documentation [39]), which makes it more likely to impact signal reception in adverse environments. Therefore, CDL-A channel model is deemed appropriate for this research to explore the potential of the neural receiver in challenging scenarios.

These channel models will take the resource grid from the transmitter as input and introduce varying levels of noise, interference, and delay to generate a noisy resource grid output for the receivers.

3.5 RECEIVERS

Chapter 2 and Section 3.2 has already covered most of the three receivers used in FIGURE 3.1, the only components that is left to discuss are the demapper and the LLRs. The demapper in the two baseline receivers computes a log-likelihood ratio (LLR) using the channel estimations from the LMMSE equalizer, whereas the neural receiver directly processes the noisy resource grid and outputs the LLR. The LLR is a soft estimate that measures the likelihood of whether the received signal is a one or a zero, determined by taking the logarithm of the ratio of the probability that the received signal is one to the probability that it is zero. The LLR is a critical element used by the subsequent decoder to make hard decisions the received signal.

4 EXPERIMENTS ON THE NEURAL RECEIVER

4.1 OVERVIEW

This chapter details experiments conducted on a simple AWGN model to gain insights into the neural network's hyper-parameters. The choice of using an AWGN model as a starting point was deliberate, as it is relatively faster to train compared to more complex models such as OFDM. This enables the effects of changing a specific hyper-parameter to be studied more quickly. For instance, while it may take 4-8 hours of training to see results for an OFDM model, an AWGN model may only take 1-2 hours. Therefore, starting with the AWGN model allows for more efficient exploration of the impact of individual hyper-parameters.

The experiments will utilize the same setup described in [12] as the default setup, which features a ResNet model with four residual blocks, a fixed learning rate, and a fixed batch size. All experiments will be trained using 50,000 iterations using the 256-QAM modulation (eight bits per symbol). In addition, due to the time-consuming nature of training deep neural networks and limited hardware resources, the experiment in this Chapter will only focus on three key aspects that extend from the default setup. The changes will be tested independently, with only one hyper-parameter altered at a time to ensure clarity and prevent confusion. Specifically, the experiments will investigate:

- ✧ the impact of implementing a cyclical decaying batch size (from inspiration in Section 2.5), and compare its performance against the default model that uses a fixed batch size.
- ✧ the performance of adding more residual blocks to the model, beyond the default setup, to determine which number of residual blocks produces the highest performance.
- ✧ the impact of using different types of learning rate schedules, including exponentially decaying learning rate, constant learning rate, and learning rate with warm restarts, while using the optimal number of residual blocks derived from the previous bullet point.

All experiment uses a conventional baseline that relies on perfect CSI to benchmark the performance. Additionally, the final section of this chapter will visually and qualitatively compare the trained constellations of 256-QAM from all experiments to evaluate of how some settings may perform better than others. TABLE 4.1 provides a comprehensive view of the parameters that will be experimented with the AWGN channel.

TABLE 4.1 PARAMETERS OF THE EXPERIMENT ON AWGN CHANNEL

Parameters	**Specifications**
Batch size	Constant / Cyclical decaying
Number of residual blocks	4/6/8
Leaning rate	Constant / Exponentially decaying / Warm restart

4.2 USING CYCLICAL DECAYING BATCH SIZE

For the fixed batch size, a batch size of 512 is used. This means that the neural network will process 512 samples in parallel during each iteration. As for implementing the cyclical decaying batch size, an array containing three power-of-two values will be initialized at the beginning. During each iteration, a random value from this array will be selected as the number of samples processed in parallel. After every 10,000 iterations, the largest value in the array will be removed, and a value that is half of the smallest value in the array will be added to the array.

For example, suppose the three values in the array are initially set to {256, 512, 1024}. During the first 10,000 iterations, each iteration will use a randomly selected batch size from these three values. After the 10,001st iteration, 1024 will be removed from the array, and 128 will be added to the array. Thus, the updated array will be {128, 256, 512}. This process will continue until the 40,001st iteration when the array will eventually become {16, 32, 64}. The simulation result is presented in FIGURE 4.1.

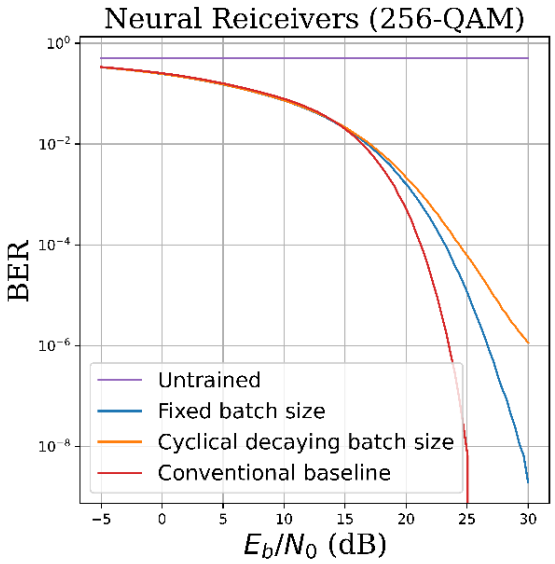


FIGURE 4.1: *Plot showing the comparison between the BER performance of neural network that is trained using a fixed batch size and cyclical decaying batch size*

The results from the graph indicate that the use of cyclical decaying batch size may not be optimal for training the neural network. One possible explanation for this is that the neural network typically requires a large number of parallel samples to learn from and extract features from the dataset effectively. When the batch size is too small, such as when the cyclical decaying batch size reaches 30,000-40,000 iterations, the neural network may not be receiving enough samples to learn from, resulting in an underfitted model. This means that the dataset is not large enough for the neural network to fully learn the underlying patterns, leading to a saturated accuracy.

4.3 USING MORE RESIDUAL BLOCKS

Extending from the default model's four residual blocks, this section investigates the impact of increasing the number of residual blocks to six and eight in the ResNet. This section will end with a qualitative analysis of the resulting performance. As illustrated in FIGURE 2.9, each residual block consists of two convolutional layers, meaning that adding one residual block is equivalent to adding two convolutional layers to the network. Generally, increasing the number of layers in a neural network widens the network, allowing data to pass through more weights and biases, and potentially enabling the network to extract more features from the input dataset. The performance of ResNet models with four, six, and eight residual blocks is presented in FIGURE 4.2.

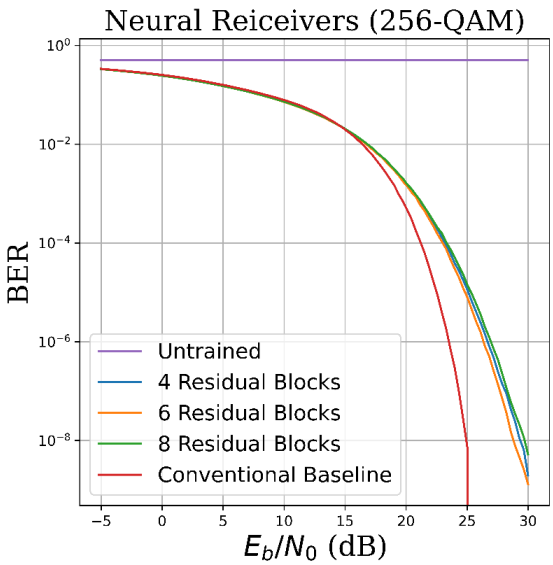


FIGURE 4.2: *Plot showing the comparison between the BER performance of neural network that uses 4, 6 and 8 residual blocks*

The results in FIGURE 4.2 show that adding more residual blocks to the ResNet can improve performance up to a certain point. Specifically, when the number of residual blocks increased from four to six, there was a slight improvement in performance. However, adding even more residual blocks (i.e.,

eight) resulted in a decrease in performance. This may be due to the risk of overfitting, where the neural network becomes too deep and memorizes the training data instead of learning the underlying patterns. This is a common problem when there are too many layers in a neural network. In the case of the AWGN channel, which is a relatively simple channel with only gaussian noise being added, having eight residual blocks (equivalent to 16 hidden layers) may be unnecessary and could lead to overfitting.

4.4 USING DIFFERENT LEARNING RATE SCHEDULES

Building on the previous section, this section aims to investigate the impact of various learning rate schedules available in the Keras API using six residual blocks in the ResNet. Specifically, this section will implement two learning rate schedulers presented in [16]: exponentially decaying learning rate and learning rate with warm restarts. The performance of these schedulers will be compared to the default setup, where a constant learning rate is employed.

To ensure optimal performance of the two learning rate schedulers, a LR range test mentioned in [17] is conducted to determine the appropriate boundaries for the learning rate. This involved utilizing a linearly increasing learning rate scheduler over 5,000 iterations, gradually increasing the learning rate from 0 to 0.01. The resulting plot of the loss against learning rate can be seen in FIGURE 4.3. It can be seen that when the learning rate is lower than 3×10^{-5} the loss reduces slowly, indicating a slow convergence. After the learning rate increase beyond 6×10^{-4} the loss began to fluctuate largely indicating occasional divergence. In between these values the loss is reducing at a reasonable rate. As a result, the base and maximum learning rate values that will be used for learning rate with restart is determined to be 3×10^{-5} and 6×10^{-4} respectively. As for the exponential decaying learning rate scheduler, the learning rate exponentially decrease from 6×10^{-4} to 3×10^{-5} throughout 50,000 iterations. The constant learning rate will use a value of 0.001 throughout all iterations.

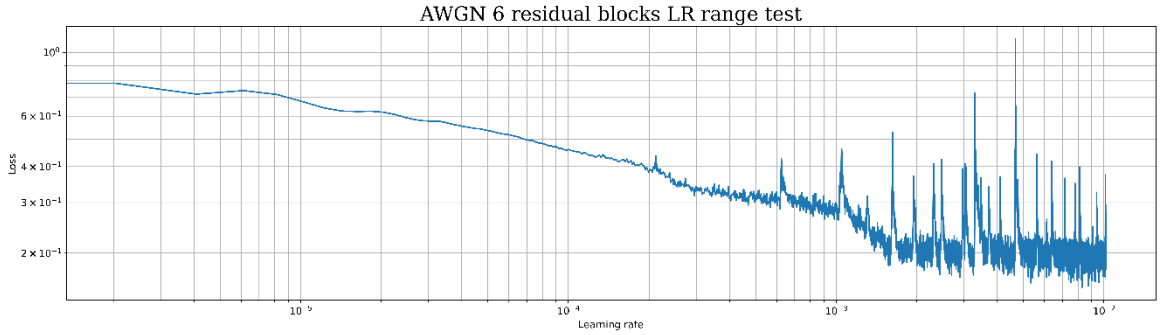


FIGURE 4.3: *A loss vs learning rate plot showing how the model's loss reacts to linearly increasing learning rate for AWGN channel*

The results of the three learning rate schedules are presented in FIGURE 4.4. According to [16], using varying learning rates can improve performance compared to a constant learning rate. This observation is supported by the results shown in FIGURE 4.4. In particular, the learning rate with warm restart outperforms the exponential learning rate, possibly because the cyclic resetting of the learning rate allows the neural network to explore different regions of the optimization landscape, potentially escaping from local minima. In contrast, the exponentially decaying learning rate may get stuck in local minima once the model enters them, as there are no large steps to help it escape.

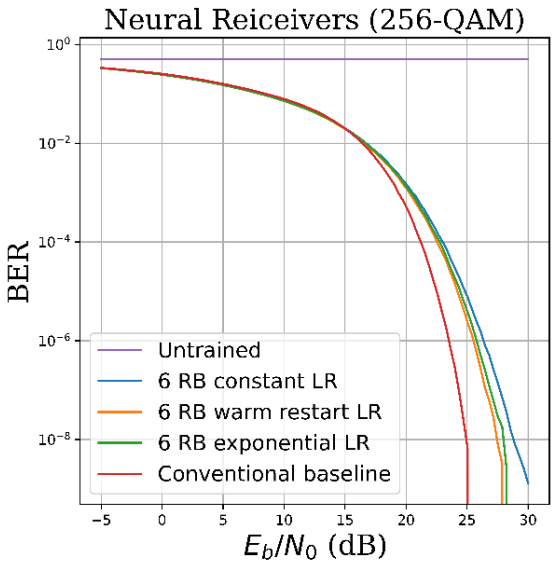


FIGURE 4.4: *Plot showing the comparison between the BER performance of neural network that constant learning rate, learning rate with warm restart and exponentially decaying learning rate.*

4.5 COMPARISON OF TRAINED CONSTELLATIONS

This section presents a comparison of six trained constellations obtained from the experiments conducted in previous sections. In all the constellation plots, the blue dots represent the original, untrained constellation, while the orange dots represent the constellation points after training. The constellations can be seen in FIGURE 4.5 and are categorized as follows. (Note: RB, LR and BS are short for residual blocks, learning rate and batch size, respectively)

- ✧ 4 RB, fixed LR, fixed BS (FIGURE 4.5*a*): This is the constellation of the default setup from 41[12].
- ✧ 4 RB, fixed LR, cyclical BS (FIGURE 4.5*b*): This is the constellation of the model that extends the default setup by implementing cyclical decaying batch size.
- ✧ 6 RB, fixed LR, fixed BS (FIGURE 4.5*c*): This is the constellation of the model where two additional residual blocks were added to the default setup.
- ✧ 8 RB, fixed LR, fixed BS (FIGURE 4.5*d*): This is the constellation of the model where four

more residual blocks were added to the default setup.

- ✧ 6 RB, exponential LR, fixed BS (FIGURE 4.5*e*): This is the constellation of the model with six residual blocks that uses exponentially decaying learning rate.
- ✧ 6 RB, warm restart LR, fixed BS (FIGURE 4.5*f*): This is the constellation of the model with eight residual blocks that uses warm restart learning rate.

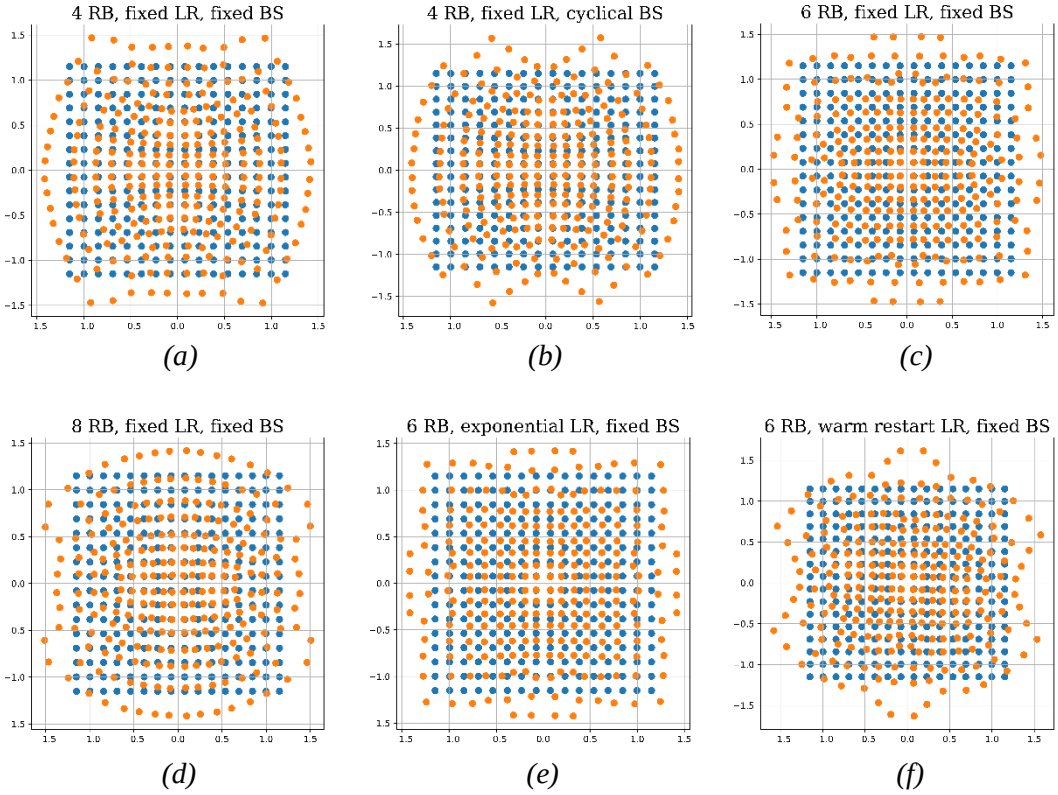


FIGURE 4.5: *Six trained constellations plotted against the original 256-QAM constellation. (a) 4 RB, fixed LR, fixed BS (b) 4 RB, fixed LR, cyclical BS (c) 6 RB, fixed LR, fixed BS (d) 8 RB, fixed LR, fixed BS (e) 6 RB, exponential LR, fixed BS, (f) 6 RB, warm restart LR, fixed BS Note: RB, LR and BS are short for residual blocks, learning rate and batch size, respectively*

(Note: the rest of this section uses A, B, C, D, E and F to represent the constellations in FIGURE 4.5)

The first two constellations to be compared are A and B, which correspond to fixed batch size and cyclical decaying batch size, respectively. Upon examination, it is clear that the constellation points at the very top and bottom corners are further apart in A compared to B. This indicates that some symbols positioned at the top and bottom corners in b may be more susceptible to interference, which could explain why the cyclical decaying batch size performs worse than the fixed batch size.

Moving on, the second comparison involves constellations A, C, and D, which represent the cases where 4, 6, and 8 residual blocks are used, respectively. Upon close inspection, it can be observed that A and D are very similar when one of them is rotated ninety degrees but not identical. This suggests

that their performance is also similar. However, C is noticeably different from both A and D. In C, there are four constellation points at the top and bottom that exclusively occupy a single row. These isolated points create more space for other constellation points to locate further apart from each other, making them more resistant to noise. This is why the model with 6 residual blocks performs better than the cases where 4 or 8 residual blocks are used.

In the final comparison, the constellations C, E, and F are compared, which correspond to models with 6 residual blocks using a constant learning rate, an exponentially decaying learning rate, and a learning rate with warm restart, respectively. Constellations C and E exhibit similar characteristics, with E being an expanded version of C. This is likely due to the fact that both constant and exponentially decaying learning rates do not involve a rise in learning rate, causing the models being stuck in the same local minimum and resulting in similar constellation structures. However, E has a better performance due to the exponentially decaying learning rate enabling the model to converge deeper with smaller learning rates at later iterations. In contrast, constellation F has a distinct structure, which can be attributed to the warm restart feature allowing the learning rate to increase every cycle and giving the model an opportunity to escape local minima and find better constellation point combinations, resulting in higher performance.

As a result, the following conclusions can be made from the experiments:

- ✧ The use of cyclical decaying batch size does not enhance the model's performance. In fact, it reduces the model accuracy as the network does not receive sufficient data to learn from.
- ✧ Utilizing a varying learning rate can improve the performance of the communication system compared to a constant learning rate. Particularly, using a learning rate with warm restarts yields better BER performance than using an exponentially decaying learning rate due to the cyclical rise in learning rate that helps the model to escape local minimums.
- ✧ Increasing the number of residual blocks (or convolutional layers) in the neural network can lead to better results. However, excessive use of residual blocks may cause overfitting of the model, which reduces its accuracy.

Based on the conclusions drawn from the experiments, the subsequent research that uses advanced 3GPP channel models will employ a large and fixed training batch size with a learning rate with warm restart scheduler. Since the CDL and UMi models are more complex and have more features than the AWGN channel, the number of residual blocks will be experimented again, with a higher number of residual blocks. Further details will be discussed in the next chapter.

5 OFDM TRAINING AND SIMULATION RESULTS

5.1 OFDM SYSTEM SIMULATION PARAMETERS

TABLE 5.1 PARAMETERS OF OFDM SYSTEM

Parameters	Specifications
QAM order	1024
Type of channel	CDL-A, UMi
Eb/No	0 to 15 dB
Direction	Uplink
Domain	Frequency
Cyclic Prefix	0 (not used in frequency domain simulation)
Carrier frequency	3.5 GHz
Subcarrier spacing	15 kHz
Speed	5 m/s
Delay spread	100 ns
LDPC code rate	0.5
Number of transmitter antenna	1
Number of receiver antenna	2
Number of subcarriers	152 (maximum subcarrier supported by Sionna)
Number of OFDM symbols	14

TABLE 5.1 displays the channel model parameters, which have been configured to simulate a near worst-case scenario of signal transmission with a high order modulation scheme in city centers. The subsequent sub-sections provide a detailed explanation of the rationale behind these parameter settings.

5.1.1 ENERGY PER BIT TO NOISE POWER SPECTRAL DENSITY RATIO (EB/NO)

Usually, a higher Eb/No ratio results in larger Euclidean distances between the constellation points, providing improved noise resilience. However, in this research, a lower Eb/No range (0 to 15 dB) is employed to assess the system performance of the neural receiver in relatively low Eb/No ratios, which contrasts with the previous experiments that used a range of -5 to 30 dB in an AWGN channel.

5.1.2 *DIRECTION*

In the uplink direction, data is transmitted from the user to the base station, with the user's antenna acting as the transmitter and the base station's antenna acting as the receiver. The channel impulse response signals are sampled at the Nyquist rate by the antennas at the base station. It is exactly the opposite for the downlink direction. However, as downlink requires additional setup in the end-to-end system, therefore this research focuses solely on simulating the uplink direction, with the possibility of exploring the downlink direction in future work.

5.1.3 *DOMAIN AND CYCLIC PREFIX*

In addition, to maximize the data rate, this research focuses on investigating the effect of using a large number of subcarriers in the OFDM system, and therefore simulates the system in the frequency domain. As such, the use of cyclic prefix, which is a time-domain technique used to prevent inter-symbol interference by adding copies of the end of the OFDM symbol to the beginning of the symbol, is not relevant in this research and will not be used. The simulation of the OFDM system in time-domain can also be explored in future work.

5.1.4 *CARRIER FREQUENCY AND SUBCARRIER SPACING*

Using a carrier frequency of 3.5 GHz strikes a balance between capacity and coverage in 5G systems, making it a popular choice. Higher carrier frequencies offer greater data capacity but have limited coverage, while lower frequencies provide wider coverage but have lower capacity. Hence, in this research, the carrier frequency of 3.5 GHz is selected for scenarios such as city centers to achieve optimal results.

On the other hand, subcarrier spacing refers to the frequency separation between two adjacent subcarriers, this typically range from 15 kHz to 960 kHz [42]. In this research, to account for higher spectral efficiency that pack more subcarriers within a given bandwidth, a small subcarrier spacing (15 kHz) is used.

5.1.5 *SPEED*

The speed mentioned in this context refers to the speed at which the transmitter antenna moves, which is determined by the movement of the user. Based on [34], the average speed of cars in central London in 2022 is 17 km/h or 4.72 m/s. It is observed that a higher user speed leads to a higher Bit

Error Rate (BER). Thus, to simulate the movement of users in various scenarios, a speed of 5 m/s is considered in this research, accounting for users who are walking, sitting in office, and travelling in car.

5.1.6 DELAY SPREAD

During transmission, signals undergo multipath propagation due to the presence of obstacles, causing different versions of the same signal to arrive at the receiver at different times. This difference in time is known as delay, and the spread of this delay is known as delay spread. To model this effect, this research uses the "nominal delay spread" profile, which is 100 ns as specified in the Sionna documentation [35].

5.1.7 LDPC CODE RATE

The LDPC code rate is a key parameter used in the baseline system, representing the ratio of message bits to parity bits in a given message block. A lower code rate implies a higher number of parity bits and a lower number of message bits, leading to better protection and higher quality of the transmitted information. To achieve a balance between information protection and transmission quality, the code rate can be varied. In this research, a code rate of 0.5 is used as a midpoint for achieving this balance.

5.1.8 NUMBER OF TRANSMITTER AND RECEIVER ANTENNA

In this study, a Single-Input-Multiple-Output (SIMO) system is used with one transmitter and two receiver antennas. While increasing the number of receiver antennas can improve the channel's resistance to interference and noise through spatial diversity and multiplexing, this study aims to replicate a near worst-case scenario, which is why the minimum number of two receiver antennas is used.

Indeed, MIMO systems can provide even higher capacity by utilizing multiple transmitter antennas for spatial diversity, multiplexing, and beamforming. Several attempts were made in this study to implement the ResNet neural receiver in a MIMO setting. However, the result appeared to be not very promising. This is probably because the model in [12] is dedicated to SIMO systems, which makes it challenging when applied to MIMO systems. Therefore, this research takes a step back and focuses on implementing a SIMO system. The result of the attempts made on the MIMO system is presented in Chapter 6.

5.1.9 *NUMBER OF SUBCARRIERS AND OFDM SYMBOLS*

This study aims to maximize data throughput by using a large number of subcarriers. However, there is a restriction on the number of subcarriers that can be used in the resource grid due to the LDPC encoder module in Sionna. Therefore, the maximum number of subcarriers that can be used is limited to 152 in this research. The number of symbols per slot is strictly defined to be 14 in 3GPP [26] 5G New Radio standard therefore it is left unchanged.

5.2 RESNET TRAINING SETUP AND RESULT

TABLE 5.2 PARAMETERS OF THE EXPERIMENT ON UMI CHANNEL

Parameters	**Specifications**
Number of residual blocks	8/12/16
Number of training iterations	50,000/100,000
Eb/No range	0-15/10-15

Chapter 4 concluded that a learning rate with warm restart and a large, fixed batch size is beneficial for training neural networks. This setup will be used for training the ResNet with advanced channel models. However, since CDL-A and UMi channel models are more complex and contain more features than AWGN, therefore the optimal number of hidden layers for ResNet with advanced channel models will be experimented again using eight, twelve and sixteen residual blocks.

Additionally, two new parameters, as shown in TABLE 5.2, is be introduced for experimentation: the number of training iterations and the Eb/No range. Firstly, the number of iterations determines the total number of samples fed into the neural network. With a batch size of 512, 50,000 iterations result in 25.6×10^6 samples, and 100,000 iterations results in 51.2×10^6 samples. Secondly, the Eb/No range indicates the amount of noise carried by each batch of samples that is fed into the neural network. A number within the given range is randomly selected during each iteration to determine the level of noise in that batch.

Prior to training the model, a LR range test in [17] is conducted to determine the optimal boundaries for the learning rate with warm restart. FIGURE 5.1 shows that the loss remains relatively constant when the learning rate is below 1×10^{-3}, and the loss dramatically increases when the learning rate is slightly above 3×10^{-3}. Therefore, for the OFDM channel models, a base learning rate of 1×10^{-3} and a maximum learning rate of 3×10^{-3} are determined as the appropriate boundaries.

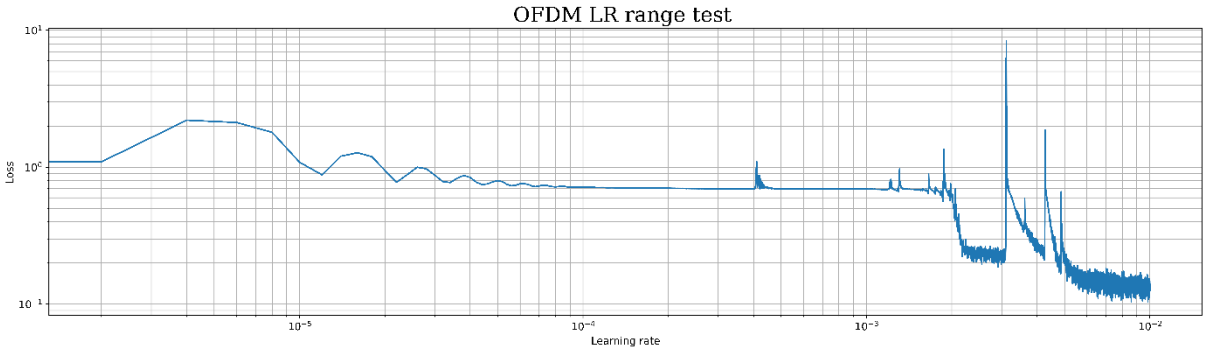
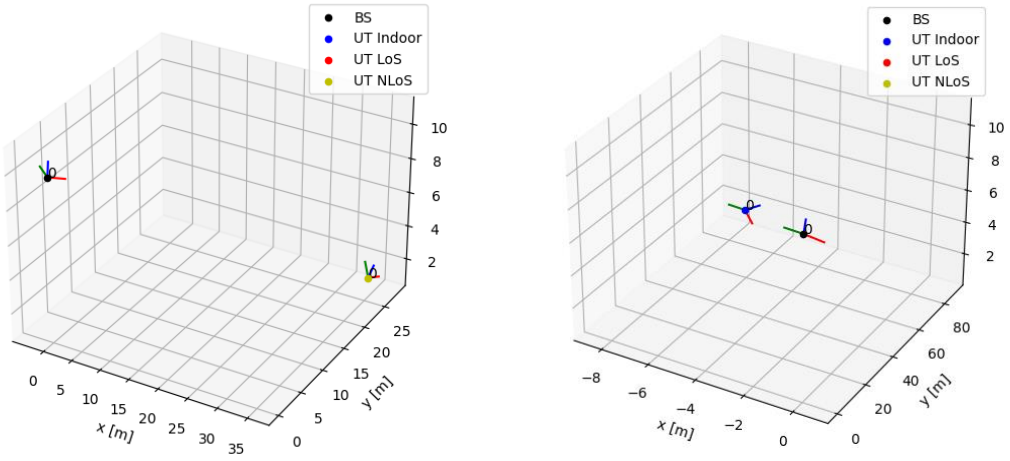


FIGURE 5.1: *A loss vs learning rate plot showing how the model's loss reacts to linearly increasing learning rate for OFDM channel*

After determining the boundaries for the learning rate with warm restart, experiments are conducted against the UMi channel model. To simulate signal transmission in UMi, a network topology (as shown in FIGURE 5.2) is required. This topology defines the position of the base station and user, their distance from each other, and signal propagation properties such as LOS, NLOS, or indoor. In this study, the generation of LOS is disabled, therefore there will only be NLOS and indoor network topologies. The example in FIGURE 5.2*a* depicts a network topology with one user terminal (UT) and one base station (BS). The signal transmission between the user antenna and the BS antenna in this case is NLOS, indicating there are obstacles between them. While the topology does not show the obstacles, they are assumed to be present. In addition, to prevent overfitting, a new network topology is generated for each iteration during the training process (shown in the location of UT and BS in FIGURE 5.2*a&b*). This randomized setup enables the neural network to learn from a diverse range of network topologies and enhances its ability to generalize to unseen data.

The number 0 next to the UT and BS indicates the index, since only one UT and BS is used in this case, both indexes are 0. The three branches on UT and BS defines their orientation.



(a) *(b)*

FIGURE 5.2: *Examples of network topologies in UMi that has one user, one base station in (a) NLOS scenario, (b) Indoor scenario*

The following subsections presents the simulation result for the ResNet neural receiver's performance on UMi and CDL-A channel models individually.

5.2.1 TRAINING RESULT FOR UMI CHANNEL MODEL

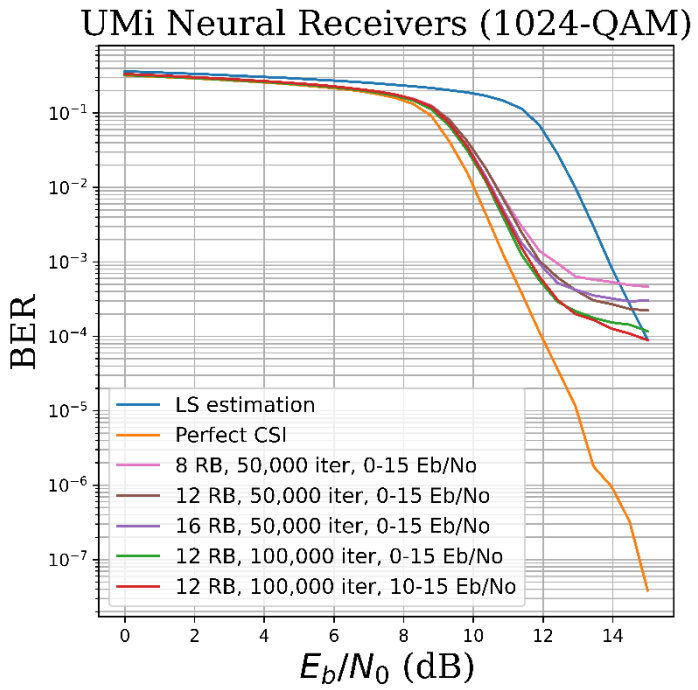


FIGURE 5.3: *BER vs Eb/No plot showing the performance of the ResNet in UMi with different setups.*

Same as in Chapter 4, only one parameter is altered at a time to ensure clarity and prevent confusion. First of all, the number of iteration is fixed to 50,000 and the Eb/No range is fixed to 0-15 dB, and the effect of increasing the number of residual blocks is examined (pink line for 8RB, dark brown line for 12 RB, and purple line for 16 RB). FIGURE 5.3 again suggests that increasing the number of residual blocks is beneficial but only up to a certain point, in this case, 12 residual blocks. When the number of residual blocks is increased 16, or equivalently 32 hidden layers, it went a bit too far and the BER has gone up since then. As a result, 12 residual blocks will be used for the subsequent analysis and simulations.

The number of iterations is the second parameter to be examined (dark brown line for 50,000 iterations, and green line for 100,000 iterations), this time the number of residual blocks (12 blocks) and the Eb/No range (0-15 dB) will remain constant. FIGURE 5.3 shows that the performance of the neural network significantly improves when it is trained on twice the amount of data. This is because a larger dataset provides the network with more information about the channel models, enabling it to better adapt to them. However, doubling the number of iterations used to train the model also doubles

the training time, from 4 to 8 hours. While increasing the number of iterations further could improve performance, this study limits the number of iterations to 100,000 due to the excessive time required beyond this point.

The final experiment explores the impact of the Eb/No range on the model's performance (green line for range of 0-15 dB, and red line for range of 10-14 dB), building upon the previous two experiments that used 100,000 iterations and 12 residual blocks. FIGURE 5.3 shows that a range of 10-15 dB yields a slight improvement in the BER at higher Eb/No value compared to a range of 0-15 dB. This suggests that when using a range of 0-15 dB may introduce batches with excessive noise, hindering the neural network's ability to learn. When the noise level is too high, it becomes difficult to identify symbols in the constellation. In contrast, a range of 10-15 dB provides more meaningful samples for the neural network to learn from.

Lastly, FIGURE 5.3 compares the performance of the neural receiver with the traditional LS estimation method. It is shown that the neural receiver achieves a significantly better BER rate at a lower Eb/No, approaching the performance of perfect CSI. However, when the Eb/No exceeds 13 dB, the neural receiver's performance appears to saturate, with the curve flattening out. This saturation suggests that the current macro setup of the neural network is achieving its optimal results with the available parameters. To further improve performance, additional parameters must be introduced to address the issue of omitted variable bias. In DL training, omitted variable bias arises when a model fails to include a relevant variable that is correlated with the important variables in the advanced channel model, which leads to the convergence of suboptimal solutions.

5.2.2 TRAINING RESULT FOR CDL-A CHANNEL MODEL

For CDL channel models, setting up a network topology is not needed. Therefore, this suggests that CDL is relatively less complex compared to UMi channel models. This can be verified by examining the perfect CSI curves in FIGURE 5.3 and FIGURE 5.4. In UMi, the perfect CSI curve reaches a BER rate of approximately 4×10^{-8} at 15 dB, whereas in CDL, the perfect CSI curve is able to reach zero BER rate at around 12.5 dB. CDL is less complex than UMi is because UMi can take into account the location of the transmitter and receiver in different scenarios, whereas CDL does not.

The latest configuration of the ResNet from subsection 5.2.1 is implemented again in the CDL-A channel model, where 12 residual blocks, 100,000 iterations and an Eb/No range of 10-15 dB is used. Similar to the case in UMi channel model, FIGURE 5.4 shows that the neural receiver outperforms the

LS estimation when the Eb/No value is less than 13 dB, and the performance of the neural receiver saturates and flattens as Eb/No increases beyond 13 dB.

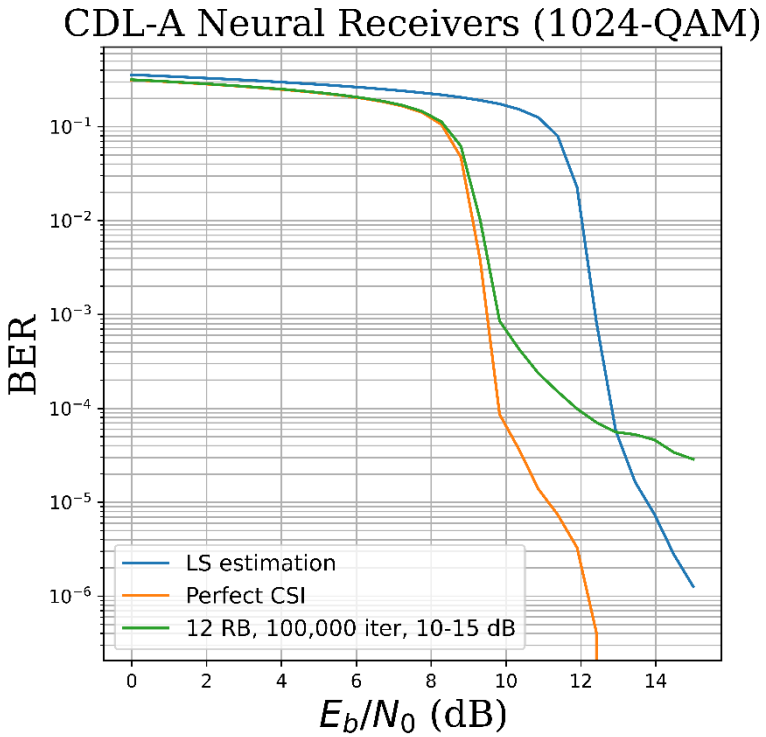


FIGURE 5.4: *BER vs Eb/No plot comparing the performance of neural receiver in CDL-A.*

5.3 RESULT DISCUSSION

The results of the performance of the neural receiver with UMi and CDL-A channel model at 1024-QAM demonstrated in Section 5.2 suggests that the neural receiver is struggling to obtain a low BER rate at high Eb/No values. Furthermore, it is shown that it is challenging for the current ResNet neural receiver to gain better results by tuning the hyper-parameters that were presented in this chapter. To address this problem, it is suggested to study, tune, and experiment with more hyper-parameters to see if the neural receiver performs better, such as adding the number of neurons in the hidden layers or adjusting the kernel size of each convolutional layer. However, given the limited time available for testing and tuning, this will be included in future work.

Apart from evaluating the performance of the ResNet neural receiver at 1024-QAM, this study also conducted additional work to assess its performance at lower-order QAM, including 4, 16, and 256-QAM. The findings indicate that the current ResNet model [12] performs significantly better at low-order QAM. This reinforces the need to introduce new parameters into the model as the current setup may be more suitable for advanced channel model at low-order QAM. For a detailed discussion of these results, please refer to Chapter 6.

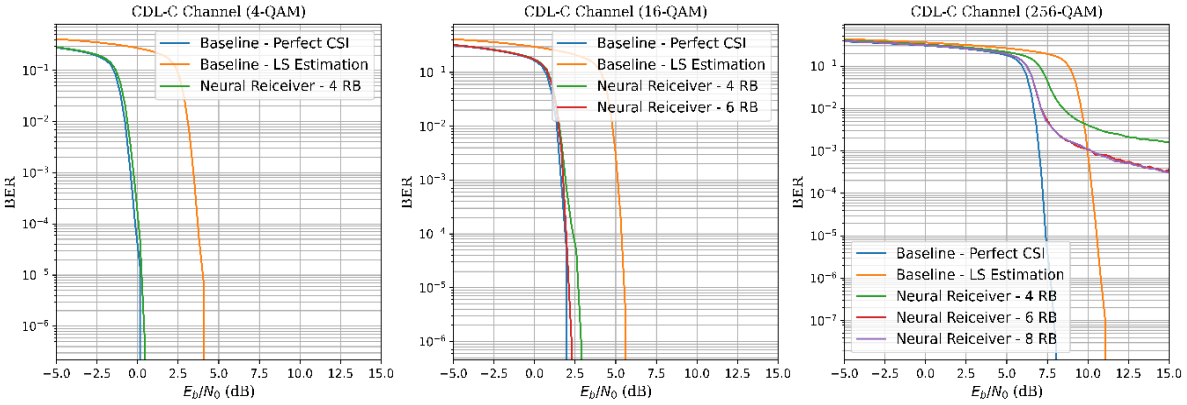
6 ADDITIONAL WORKS

This chapter provides additional works that further enhance the understanding of the ResNet neural receiver and support the conclusions drawn in Chapter 7. The first additional work assesses the performance of the ResNet receiver at different QAM orders, specifically 4, 16, and 256-QAM. The results provide valuable insights into the receiver's performance capabilities and limitations. The second additional work explores the potential of the ResNet receiver in a MIMO setup, representing an initial attempt to extend the model's capabilities to more complex communication scenarios. Both of these works provide important context and support for the conclusions presented in Chapter 7,

6.1 RESNET AT LOW-ORDER QAM

This section presents the simulation result for the assessment of neural receiver trained against CDL-C channel model at 4, 16, and 256-QAM. In this section, all models are trained with less residual blocks and 50,000 iterations to reduce computational time.

FIGURE 6.1*a* illustrates that training the neural receiver on 4-QAM (2 bits per symbol) yields highly promising results even with just four residual blocks. This is because the Euclidean distance between constellation points in low order QAM is large, making them more resilient to noise. Notably, the neural receiver's performance is nearly optimal and significantly outperforms LS estimation.



(a) *(b)* *(c)*

FIGURE 6.1: *BER vs Eb/No plot comparing the performance of neural receiver in CDL-C at (a) 4-QAM, (b) 16-QAM, (c) 256-QAM.*

Upgrading to 16-QAM modulation shown in FIGURE 6.1*b*, although the performance of the neural receiver with four residual blocks still remains promising compared to LS estimation, it begins to exhibit a slight decline at higher Eb/No. However, adding two more residual blocks to the model restores the performance to a near-optimal level. Additionally, as the QAM order increases, it can be observed that the perfect CSI curve shifts to the right due to the need to fit more constellation points into a limited space. This reduces the Euclidean distance between the points, making them less resilient to noise.

Unluckily, FIGURE 6.1*c* reveals that when the modulation scheme rises to 256-QAM, the neural receiver's performance exhibits a BER that is comparable to the results in Section 5.2. In this case, the performance is initially excellent but gradually saturates and flattens as it approaches high Eb/No. Increasing the number of residual blocks is not as effective as in FIGURE 6.1*b*, where adding six residual blocks slightly improves performance but still leads to saturation. Adding eight residual blocks does not result in any further performance gains, suggesting that the model has reached its full potential with the current setup. Thus, additional hyper-parameter tuning or model improvement is needed to enhance the receiver's performance at high Eb/No.

The first two findings of this section are promising, as the neural receiver shows superior performance compared to LS estimation at low-order QAM, achieving near-optimal performance. However, the results also demonstrate that the proposed model faces a significant challenge in maintaining consistently high performance across all Eb/No values. Specifically, the neural receiver exhibits a decline in performance as the modulation scheme increases, with 256-QAM showing saturation and flattening of the BER curve. This saturation effect indicates that the proposed model has limitations in its ability to effectively handle the increasing complexity of higher-order QAM. Thus, there is a clear need for model improvement to overcome this challenge and maintain high performance across all modulation schemes and Eb/No values. Potential solutions include hyper-parameter tuning, architectural improvements, or the incorporation of additional training data to enhance the model's ability to handle complex QAM schemes.

6.2 AN ATTEMPT OF USING RESNET IN MIMO SYSTEM

In this section, an initial attempt is made to extend the SIMO system to a MIMO system, with the aim of evaluating the ResNet neural receiver's performance at higher data rates. Three different channel models are used, namely UMi, UMa, and CDL-C. MIMO systems are known to work exceptionally well in conjunction with OFDM systems. MIMO achieves high capacity by transmitting multiple signals over multiple antennas, while OFDM divides a radio channel into numerous closely spaced subchannels to provide high throughput. This combination is a dominant feature of modern wireless

communication systems. Figure 6.2 presents the results of using the ResNet neural receiver in a 4-QAM MIMO-OFDM system (two transmitter and receiver antennas), with the use of the three channel models.

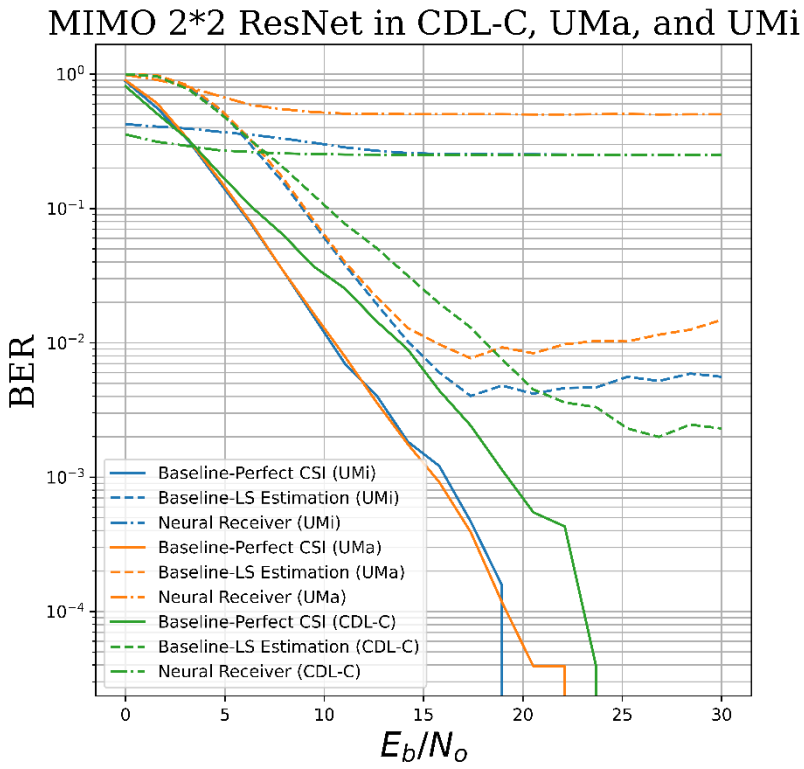


FIGURE 6.2: *BER vs Eb/No plot comparing the performance of neural receiver in MIMO system.*

Unfortunately, the results presented in Figure 6.2 reveal that the attempt to train the ResNet neural receiver in a MIMO setting was unsuccessful. The BER curve for the neural receiver remained almost completely flat throughout the entire range of Eb/No values. Despite several attempts to improve the accuracy, such as increasing the number of iterations or residual blocks, no significant improvements were observed. These results indicate that the current setup of the autoencoder is not well-suited for training a MIMO-based channel, highlighting the need for further research in this area.

One possible reason for the failure of the ResNet neural receiver in MIMO settings is the difference in the shape of the input data between MIMO and SIMO systems. The shape of the input data depends on the number of transmitter antennas used, which differs between the two systems. The current neural network is designed to accept input data with a SIMO system shape, where only one transmitter antenna is used. Therefore, it may require adjustments to the shape of the input data to enable the neural network to recognize and learn from the input data from MIMO systems.

7 CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

The main aim of thesis is to study, develop and assess the performance of an end-to-end autoencoder equipped with a ResNet-based neural receiver simulated under realistic channel models and high data rates.

The thesis began by exploring the underlying principles of deep learning, focusing on two essential algorithms that neural networks use: SGD and backpropagation. The purpose of SGD is to solve the minimization problem of the loss function, and backpropagation enables the result of SGD to be updated throughout the entire network.

Section 2.2 then focused on one of the crucial hyper-parameters of neural networks, the learning rate. This study evaluated different kinds of schedulers and conducted experiments with an AWGN channel model to draw conclusions on the best scheduler. Although the optimal scheduler CLR mentioned in [16] cannot be utilized due to limitations in Keras, this inspired further exploration of the cyclical nature of hyper-parameters, which led to another experiment using cyclical decaying batch size in training the neural network. Unfortunately, it was found that cyclical decaying batch size was not an optimal solution.

Chapter 2 also evaluated the pros and cons of different forms of neural networks, studying their application scenarios, and explaining the reasoning behind using a ResNet-based neural receiver in this research. The ResNet-based neural receiver was chosen because it effectively addresses the vanishing gradient problem in very deep neural networks.

In Chapter 3, the Sionna-based end-to-end system was introduced along with an explanation of each building block's core working principles. A flowchart was provided to illustrate the workflow, and baselines were established for benchmarking the autoencoder's performance.

After gaining a theoretical understanding of DL and communication systems, practical experience was gained by conducting insightful experiments on the AWGN channel model to tune the hyper-

parameters. The conclusions drawn from these experiments were then implemented in training the autoencoder with CDL-A and UMi channel models, which provide a more realistic representation of channel conditions. This allowed for a thorough evaluation of the performance of the ResNet-based neural receiver in more realistic channel scenarios. The results from these experiments provided valuable insights into the strengths and limitations of the proposed system and led to the discussion of further improvements in the design of the autoencoder.

Lastly, Chapter 6 presents two additional works conducted to gain more insights into the capability of the ResNet neural receiver in this study. The first work verified the exceptional capability of obtaining impressive results under advanced channel models at low-order QAM. However, it also identified the limitation of the current ResNet in consistently achieving the same level of performance at high Eb/No values. The second additional work took a first step in trying to optimize the MIMO system with autoencoders. Although the result is unsatisfying, some useful insights are gained that may lead to possible improvements in the future.

In summary, the thesis covered a range of theoretical topics including DL, end-to-end learning, differentiable communication systems, and hyper-parameter tuning. In addition, practical experiments were conducted, leading to some promising results, but also identifying several directions for further research. Overall, this work contributes to advancing the understanding of the capabilities and limitations of ResNet-based neural receivers in communication systems, and lays the groundwork for future research in this area.

7.2 FUTURE WORK

Firstly, a solution to the saturation problem in the current autoencoder setup is essential. Due to the time-consuming nature of training neural networks, this research only explored a limited number of hyperparameters such as learning rate, number of hidden layers, and iterations. One possible solution is to tune more hyperparameters such as kernel sizes in convolutional layers and the number of neurons in the hidden layer. Additionally, other forms of neural networks such as RNN and LSTM networks can be employed to learn from the time series patterns that underlie the channel models. Furthermore, the study can attempt to use larger batch sizes and more iterations to provide more data for the neural network to learn from. However, this requires better hardware, otherwise the computational time would be extensive.

In fact, many studies attempted to seek a way to find the optimal number of hidden neurons for different applications. For instance, researchers in [36] experimented with different combinations of

numbers of neurons in the hidden layer based on their mathematical derivation and demonstrated promising results for financial data mining applications. They also referred to two other papers, [37] and [38], which proposed an innovative approach that removed the need for manual tuning by using an algorithm that dynamically alters the number of hidden neurons based on training errors. In [37]'s algorithm, a threshold value is selected first, and then a neuron is added to the hidden layers once the training error rises above the threshold to account for better results. [38] is very similar to [37], but it removes a hidden neuron when the training error is too low, possibly to ensure that the neural network does not overfit the dataset. The dynamic allocation of hidden neurons is an interesting aspect that can be implemented in future work.

After resolving the saturation issue, there is potential for further improvement by introducing downlink and time domain simulations into the system through the addition or modification of certain building blocks in Sionna. This will enable evaluation of the effectiveness of the autoencoder in a complete communication system. Specifically, including the downlink direction in the simulation will transform the BS antenna into the transmitter and the UT antenna into the receiver, allowing for observation of the autoencoder's effectiveness in simulating both signal transmission and reception for BS and UT. Moreover, the addition of time domain simulations can facilitate examination of the effectiveness of autoencoder in tackling the inter-symbol interferences resulting from time-delayed symbols, where multipath signals of the same version may arrive differently with respect to time.

The final future work could explore the extension of ResNet architecture to train MIMO-OFDM systems. As discussed in Section 6.2, the incompatibility in the shape of input data is likely the reason why the current ResNet cannot train MIMO systems. To resolve this issue, the code could be manually debugged line by line for both SIMO and MIMO input data shapes, followed by comparing and studying their differences. Then, by utilizing TensorFlow, the MIMO input shape could be adjusted to become recognizable and trainable by the ResNet. If the shape of input data is not the problem, the convolutional layers setup of the ResNet should be carefully examined to identify any configuration mismatches that may have caused the incompatibility. Lastly, if both the shape of input data and the configuration of the ResNet are not the issue, it may be necessary to check the correctness of the differentiable communication blocks implemented by Sionna. Additional blocks may need to be added to the end-to-end system to resolve the incompatibility problem. If necessary, it would be helpful to print out the input data and examine the variation of the data as it goes through each individual building blocks to manually check the correctness of data. To do this, it is recommended to use a low modulation scheme such as Binary Phase Shift Keying (BPSK), a low batch size, and a low number of transmitter and receiver antennas so the shape of data is low and easier to read. These approaches can help confirm that the system is functioning correctly, identify any issues that need to be addressed, and possibly lead to a successful implementation of the autoencoder in MIMO system.

REFERENCES

[1] M. E. Tarerefa, A. K. Benjamin and W. Deinmodei, "Performance Analysis of OFDM signal using Different Modulation Schemes with Least Square Channel Estimation Technique," *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)*, Lagos, Nigeria, 2022, pp. 1-4, doi: 10.1109/NIGERCON54645.2022.9803078.

[2] R. T. Kamurthi, S. R. Chopra and A. Gupta, "Higher Order QAM Schemes in 5G UFMC system," *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)*, Pune, India, 2020, pp. 198-202, doi: 10.1109/ESCI48226.2020.9167619.

[3] NVlabs *NVlabs/Sionna: Sionna: An open-source library for next-generation Physical Layer Research*, GitHub. Available at: https://github.com/NVlabs/sionna.

[4] Team, K. *Simple. flexible. powerful., Keras*. Available at: https://keras.io/.

[5] Tensorflow, *TensorFlow*. Available at: https://www.tensorflow.org/.

[6] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563-575, Dec. 2017, doi: 10.1109/TCCN.2017.2758370.

[7] Bui, Tam and Tran, Xuan Nam and Phan, Anh Huy, *Deep Learning Based Mimo Systems Using Open-Loop Autoencoder*. Available at SSRN: https://ssrn.com/abstract=4375609 or http://dx.doi.org/10.2139/ssrn.4375609.

[8] L. Pellatt, M. Nekovee and D. Wu, "A Concurrent Training Method of Deep-Learning Autoencoders in a Multi-user Interference Channel," *2021 17th International Symposium on Wireless Communication Systems (ISWCS)*, Berlin, Germany, 2021, pp. 1-6, doi: 10.1109/ISWCS49558.2021.9562181.

[9] M. Honkala, D. Korpi and J. M. J. Huttunen, "DeepRx: Fully Convolutional Deep Learning Receiver," in *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3925-3940, June 2021, doi: 10.1109/TWC.2021.3054520.

[10] Y. Zhu, C. Gong, J. Luo, M. Jin, X. Jin and Z. Xu, "Indoor Non-Line of Sight Visible Light Communication with a Bi-LSTM Neural Network," *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/ICCWorkshops49005.2020.9145317.

[11] M. Stark, F. Ait Aoudia and J. Hoydis, "Joint Learning of Geometric and Probabilistic

Constellation Shaping," *2019 IEEE Globecom Workshops (GC Wkshps)*, Waikoloa, HI, USA, 2019, pp. 1-6, doi: 10.1109/GCWkshps45667.2019.9024567.

[12] Sionna *Neural Receiver for OFDM SIMO Systems* - Sionna 0.14.0 documentation. Available at: https://nvlabs.github.io/sionna/examples/Neural_Receiver.html.

[13] Nielsen, M.A. (1970) *Neural networks and deep learning*. Determination Press. Available at: http://neuralnetworksanddeeplearning.com/index.html.

[14] Yunhuijang (2022) *Gradient descent (steepest descent) 설명 (learning rate 설명, gradient descent 의 한계점), 유니의 공부*. TISTORY. Available at: https://process-mining.tistory.com/175.

[15] *Representation for high learning rate α and low learning rate α.* Available at: https://www.researchgate.net/figure/Representation-for-high-learning-rate-a-and-low-learning-rate-a_fig1_338139807.

[16] J. Konar, P. Khandelwal and R. Tripathi, "Comparison of Various Learning Rate Scheduling Techniques on Convolutional Neural Network," *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, Bhopal, India, 2020, pp. 1-5, doi: 10.1109/SCEECS48394.2020.94.

[17] Smith, Leslie N. "Cyclical Learning Rates for Training Neural Networks." *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2015): 464-472.

[18] Rosebrock, A. (2021) *Cyclical learning rates with Keras and deep learning, PyImageSearch*. Available at: https://pyimagesearch.com/2019/07/29/cyclical-learning-rates-with-keras-and-deep-learning/.

[19] Hoffmann, M. (2017) *Exploring stochastic gradient descent with restarts (SGDR), Medium. Medium*. Available at: https://markkhoffmann.medium.com/exploring-stochastic-gradient-descent-with-restarts-sgdr-fa206c38a74e.

[20] Pechyonkin, M. (2018) *Stochastic weight averaging - a new way to get state of the art results in Deep learning, Medium. Towards Data Science.* Available at: https://towardsdatascience.com/stochastic-weight-averaging-a-new-way-to-get-state-of-the-art-results-in-deep-learning-c639ccf36a.

[21] *Learning rate of exponential decay | download scientific diagram*. Available at: https://www.researchgate.net/figure/Learning-rate-of-exponential-Decay_fig3_363817817.

[22] Y. Zhu, C. Gong, J. Luo, M. Jin, X. Jin and Z. Xu, "Indoor Non-Line of Sight Visible Light Communication with a Bi-LSTM Neural Network," *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/ICCWorkshops49005.2020.9145317.

[23] Silpa C, Vani A, Naidu KR. (2023) Deep Learning Based Channel Estimation for MIMO-OFDM System with Modified ResNet Model. *Indian Journal of Science and Technology*. 16(2): 97-108. https://doi.org/10.17485/IJST/v16i2.2154

[24] *Part 4: Toward learned receivers - Sionna 0.14.0 documentation*. Available at: https://nvlabs.github.io/sionna/examples/Sionna_tutorial_part4.html.

[25] F. Ait Aoudia and J. Hoydis, "End-to-End Learning for OFDM: From Neural Receivers to Pilotless Communication," in *IEEE Transactions on Wireless Communications*, vol. 21, no. 2, pp. 1049-1063, Feb. 2022, doi: 10.1109/TWC.2021.3101364.

[26] *The Mobile Broadband Standard* 3GPP. Available at: https://www.3gpp.org/.

[27] O. Simeone, "A Very Brief Introduction to Machine Learning with Applications to Communication Systems," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648-664, Dec. 2018, doi: 10.1109/TCCN.2018.2881442.

[28] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis and S. ten Brink, "Trainable Communication Systems: Concepts and Prototype," in *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5489-5503, Sept. 2020, doi: 10.1109/TCOMM.2020.3002915.

[29] A. Felix, S. Cammerer, S. Dörner, J. Hoydis and S. Ten Brink, "OFDM-Autoencoder for End-to-End Learning of Communications Systems," *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Kalamata, Greece, 2018, pp. 1-5, doi: 10.1109/SPAWC.2018.8445920.

[30] H. Ye, G. Y. Li and B. -H. Juang, "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems," in *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114-117, Feb. 2018, doi: 10.1109/LWC.2017.2757490.

[31] Z. Zhao, M. C. Vuran, F. Guo and S. D. Scott, "Deep-Waveform: A Learned OFDM Receiver Based on Deep Complex-Valued Convolutional Networks," in *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2407-2420, Aug. 2021, doi: 10.1109/JSAC.2021.3087241.

[32] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[33] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[34] *London is the world's slowest city: TomTom newsroom*, TomTom. Available at: https://www.tomtom.com/newsroom/explainers-and-insights/london-is-the-worlds-slowest-city/.

[35] *Wireless - Sionna 0.14.0 documentation.* Available at: https://nvlabs.github.io/sionna/api/channel.wireless.html?highlight=delay+spread#sionna.channel.tr38901.CDL.delay_spread.

[36] Xu, Shuxiang and Ling Chen. "*A novel approach for determining the optimal number of hidden layer neurons for FNN's and its application in data mining*." (2008).

[37] T. Ash, "Dynamic node creation in backpropagation networks," *International 1989 Joint Conference on Neural Networks*, Washington, DC, USA, 1989, pp. 623 vol.2-, doi: 10.1109/IJCNN.1989.118509.

[38] Y. Hirose, K. Yamashita and S. Hijiya, "Backpropagation algorithm which varies the number of hidden units," *International 1989 Joint Conference on Neural Networks*, Washington, DC, USA, 1989, pp. 625 vol.2-, doi: 10.1109/IJCNN.1989.118518.

[39] 3GPP TR 38.901 version 17.0.0 Release 17, *Work programme - work item detailed report*. Available at: https://portal.etsi.org/webapp/workprogram/Report_WorkItem.asp?WKI_ID=65119.

[40] S. Khan, K. S. Khan and S. Y. Shin, "Symbol Denoising in High Order M-QAM using Residual learning of Deep CNN," *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2019, pp. 1-6, doi: 10.1109/CCNC.2019.8651830.

[41] Y. Yu, J. Ying, P. Wang and L. Guo, "A data-driven deep learning network for massive MIMO detection with high-order QAM," in *Journal of Communications and Networks,* vol. 25, no. 1, pp. 50-60, Feb. 2023, doi: 10.23919/JCN.2022.000055.

[42] *5GShareTechnote* Available at: https://www.sharetechnote.com/html/5G/5G_FrameStructure.html.

APPENDIX A: SOFTWARE USED

Filename **Algorithm** **Package**	**Supplier** **Source** **Author website**	**Use** **Modification code** **Student written**
Python	https://www.python.org/	Used for building the auto-encoder, running simulations, and plotting graphs
Visual Studio Code	https://code.visualstudio.com/	Used to execute codes written in Python
Sionna	https://nvlabs.github.io/sionna/	Used for simulating physical layer of wireless communications
Keras	https://keras.io/	Used for training the neural networks
Tensorflow	https://www.tensorflow.org/	Used as a backend for Keras to train neural networks, also used for manipulating the shape of input data sets
NumPy	https://numpy.org/	Used for creating array of integers for different data requirements
pickle	https://docs.python.org/3/library/pickle.html	Used for converting Python objects into a byte stream to store it in a file
matplotlib	https://matplotlib.org/	Used for plotting simulation results
awgn_model.py	https://nvlabs.github.io/sionna/examples/Sionna_tutorial_part2.html	Used for training models and running simulations in an AWGN channel. Modified to fit the aim of this research
ofdm_model.py	https://nvlabs.github.io/sionna/examples/Neural_Receiver.html	Used for training models and running simulations in an OFDM channel. Modified to fit the aim of this research

awgn_plot.py	Student written	Used for plot the simulation result obtained from awgn_model.py
ofdm_plot.py	Student written	Used for plot the simulation result obtained from awgn_model.py
lr_range_test_plot.py	Student written	Used to plot the simulation result for the LR range test
q4.mlx	Student written	Used to plot the 3D representation of the peak function and the Eggholder function in MATLAB