IMPROVING CHANNEL EQUALIZATION WITH NEURAL NETWORK AND REINFORCEMENT LEARNING

By
Quyet Nguyen

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2021

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Master's Committee, we certify that we have read the thesis prepared by Quyet Van Nguyen, titled Improving Channel Equalization with Neural Network and Reinforcement Learning and recommend that it be accepted as fulfilling the dissertation requirement for the Master's Degree.

Tamal Bose
_____ Date: August 18, 2021
Tamal Bose

_____ Date: August 18, 2021
Gregory Ditzler

Ratchaneekorn Thamvichai
_____ Date: August 18, 2021
Ratchaneekorn Thamvichai

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to the Graduate College.

I hereby certify that I have read this thesis prepared under my direction and recommend that it be accepted as fulfilling the Master's requirement.

Tamal Bose
_____ Date: August 18, 2021
Tamal Bose
Master's Thesis Committee Chair
Electrical and Computer Engineering

ACKNOWLEDGEMENTS

I would like to use this opportunity to give a special thank to Dr. Tamal Bose for his support and guidance throughout the course of this work. With his vision, Dr. Bose has made my research process exciting, fun and challenging. His experience in the research field motivated me to go further and set the goal higher as the process goes on.

I would also like to give a special thank to Noel Teku, a Ph.D. student in our lab, and Dr. Kay, a mentor in our lab. Noel has been also a great guidance and helps any time I needed him to. He has spent so much time giving me direction as well as insight toward the proper way to conduct research. I am very grateful to him, and I can not imagine going thru this process without his guidance. Dr. Kay has provided feedbacks for all the works that lead to this thesis. Her experience in the field has been a great resource for the completion of this thesis.

Finally, I would like to thank my family and friends who have supported me. There was a time when research became stressful, they were there to provide me with comfort and motivation to continue with my work.

Table of Contents

List of Figures ..…….. 6
List of Tables …….. 8
List of Abbreviations …… 9
Abstract ……..10

Chapter 1: Introduction ……...12
1.1 Project Overview ……12
1.2 Literature Review and Related Works…….14
1.2.1 Neural Network for Channel Equalization ……..14
1.2.2 Epsilon Greedy Algorithm for Neural Network Tuning ……..15
1.3 Neural Network Equalizer Overview ……16
1.4 NN Equalizer Creates with Karas in TensorFlow Platform ...……..17
1.5 Thesis Structure ……..18

Chapter 2: Neural network (NN) equalization in detail ……20
2.1 NN equalizer using Multilayer Perceptron Architecture ……20
2.1.1 Simulation Set up ……20
2.1.2 Results and Discussion ……..23
2.2 NN Equalizer using Convolutional Architecture ……27
2.2.1 Convolution Neural Network Architecture ……...27
2.2.2 How CNN is used for Channel Equalization ……28
2.2.3 Simulation Set up ……28
2.2.4 Results and Discussion ……...29
2.3 Chapter Summary ……..32

Chapter 3: System of Neural Network Equalizers ……..33
3.1 Why using System of Neural Network Equalizers ……...33
3.2 How the System Works ……33
3.3 Applying System of NNs to Equalize Selected channels ……35
3.3.1 Simulation Set up ……35

3.3.2 Results and Discussion ……...36
3.4 Applying System of NNs to Equalize HF Channel ……...41
3.4.1 Simulation Set up ……41
3.4.2 Results and Discussion ……...41
3.5 Chapter Summary ……..47

Chapter 4: Tuning Neural Network Using Epsilon Greedy Algorithm ……...48
4.1 Introduction to Epsilon Greedy Algorithm ……48
4.1.1 Reinforcement Learning ……48
4.1.2 Epsilon Greedy Algorithm ……49
4.2 NN Equalizer tuned with Epsilon Greedy Algorithm ……..49
4.3 Results and Discussion ……..51
4.3.1 Simulation Set up ……51
4.3.2 Results and Analysis ……...53
4.4 Chapter Summary ……..54

Chapter 5: Conclusion and Future Works ……56

References ……..58

List of Figures

Figure 1: data structure for the input layer of an MLP neural network equalizer [7] has used for their simulation……21
Figure 2: MLP neural network architecture used for this work……23
Figure 3: communication system ……...23
Figure 4: MLP neural network equalizer on FIR filter channel……..24
Figure 5: MLP neural network equalizer on HF good channel condition……25
Figure 6: MLP neural network equalizer on HF good moderate condition…….25
Figure 7: MLP neural network equalizer on HF good poor condition……26
Figure 8: Error Performance vs SNR of an adaptive DFE from [43].……26
Figure 9: general structure of a convolution neural network……27
Figure 10: data restructure for the convolution neural network…….....28
Figure 11: CNN structure used for this work……...29
Figure 12: comparison between CNN and MLP equalizer on FIR filer channel……...30
Figure 13: compare between CNN and MLP equalizer on HF good channel condition……31
Figure 14: compare between CNN and MLP equalizer on HF moderate channel condition……31
Figure 15: compare between CNN and MLP equalizer on HF poor channel condition……32
Figure 16: System of neural network equalizers……..34
Figure 17: NN equalizer trained on linear phase channel and test on channel that varies over time……..37
Figure 18: NN equalizer trained on squared channel and test on channel that varies over time……..38
Figure 19: NN equalizer trained on flat fading channel and test on channel that varies over time……..38
Figure 20: NN equalizer trained on Cubic channel and test on channel that varies over time……..39
Figure 21: NN equalizer trained on Tanh channel and test on channel that varies over time……..39
Figure 22: system of neural network equalizers vs NN equalizer trained on linear phase channel, both test on channel that varies over time…….40
Figure 23: NN equalizer train on HF good condition and test on HF with condition change from good to moderate to poor……43
Figure 24: NN equalizer train on HF moderate condition and test on HF with condition change from good to moderate to poor……43
Figure 25: NN equalizer train on HF poor condition and test on HF with condition change from good to moderate to poor……44
Figure 26: system of NN equalizers vs NN equalizer train on HF good channel, both test on HF channel that varies from good to moderate to poor……44
Figure 27: NN equalizer train on HF good condition and test on HF with condition change from poor to moderate to good……...45
Figure 28: NN equalizer train on HF moderate condition and test on HF with condition change from poor to moderate to good……...45

Figure 29: NN equalizer train on HF poor condition
and test on HF with condition change from poor to moderate to good……...46
Figure 30: system of NN equalizers vs NN equalizer train on HF good
channel, both test on HF channel that varies from poor to moderate to good……46
Figure 31: Pseudocode of epsilon greedy strategy……...49
Figure 32: BER vs SNR of Fixed NN equalizer and epsilon greedy
tuned NN equalizer on HF good condition……..53
Figure 33: BER vs SNR of Fixed NN equalizer and epsilon greedy
tuned NN equalizer on HF moderate condition……..54
Figure 34: BER vs SNR of Fixed NN equalizer and epsilon greedy
tuned NN equalizer on HF poor condition……...54

List of Tables

Table 1: HF channel condition based on delay spread and Doppler spread …….20

List of Abbreviations

BER	Bit Error Rate
ANN	Artificial Neural Network
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase-Shift Keying
DDE	Decision Directed Equalizer
DFE	Decision Feedback Equalizer
FIR	Finite Impulse Response
HF	High Frequency
ISI	Inter-symbol Interference
MHz	MegaHertz
ML	Machine Learning
MMSE	Minimum Mean-Square Error
RL	Reinforcement Learning
SNR	Signal to Noise Ratio
MLP	Multilayer Perceptron
RBF	Radical Basic Function
LSTM	Long Short-Term Memory

ABSTRACT

In wireless communications, transmitted signals suffer from distortion caused by the channel. Equalization is used to mitigate such effects. However, a receiver needs to be adaptable to account for many types of channel effects, some of which may be nonlinear. Neural networks have been proven to be an effective method for wireless channel equalization due to their ability to learn and solve complex problems. In this thesis, two different techniques are presented to improve the use of Artificial Neural Networks (ANN) such that they can perform channel equalization more effectively. For the first technique, we present a system consist of multiple neural network (NN) equalizers, each trained on a specific channel such that a signal can still be equalized regardless of changes in channel conditions. In the first part of this experiment, we test the performance of this system of neural network equalizers on arbitrary channel models (i.e. squared, cubic, etc.). In the second part, we investigate the effectiveness of the system of neural network equalizers for different High Frequency (HF) channel conditions. The output of each NN equalizer in our proposed system is combined and optimized to select the best-equalized signal. Past research in the literature indicates that NN equalizers are vulnerable when channel conditions change because NN equalizers are trained on specific channel conditions. Based on simulation results, we show that our system can learn which NN equalizer is best suited for a particular channel as the channel varies over time. For the second technique to improve the channel equalization performance, we proposed using reinforcement learning to tune the hyperparameters of a neural network equalizer. When a neural network is created, before it can be deployed, a process called hyperparameter tuning is required for a neural network to perform at its best at a given application. For this work, we used an annealing epsilon greedy algorithm, which is a reinforcement learning technique to tune different attributes of a neural network equalizer. Reinforcement learning has been used to tune neural networks for other applications, but to the best of our knowledge, it has not been done for neural network equalizers. HF is also the assumed channel for this part of our work, and we investigate the effectiveness of using the annealing epsilon greedy algorithm to tune a neural network equalizer by

comparing its equalization performance with a fixed neural network equalizer (i.e. by fixed we mean that the neural network can learn the weights but cannot change the value of its hyper-parameters). From the results obtained, at three distinct HF channel conditions used for this work, the neural network equalizer tuned with the epsilon greedy algorithm can outperform a fixed neural network equalizer.

CHAPTER 1
INTRODUCTION

1.1PROJECT OVERVIEW

The primary purpose of this thesis is to investigate the effectiveness of using neural networks to perform channel equalization. Neural networks have been introduced since 1950, the mathematical complexity of a neural network was massively overwhelmed by the computing power at the time. Until recent times, the development of computer chips has made neural network applications became feasible [23]. Within the area of communication systems, neural networks have been implemented to perform several tasks such as equalization, channel estimation, and so on... The focus of this project is toward the application of neural networks for channel equalization. In a typical communication system, when the signal is transmitted over a communication channel, the channel will introduce impairment to the transmitted signal in the form of intersymbol interferences or ISI [24]. An equalizer is implemented at the receiver to remove distortion from the transmitted signal [25]. With its capability of learning complex and non-linear problems which are applied to the task of an equalizer, the neural network can then be used as a channel equalizer [17]. Neural network equalizer has become more popular in the research community due to its potential of achieving a performance gain. Literature indicates that the equalization performance of a neural network is superior to the conventional equalization techniques such as DFE, DDE, zeros forcing equalizer, and linear equalizer [1], [2], [4], [6], [10], and references therein.

The purpose of this project is to be able to construct a neural network, or in some cases, a system of neural networks to further improve the performance of the channel equalizer. The communication channel model is a mathematical model that mimics a channel that existed in the real world [26], which can then be used to evaluate a new technique to improve the existing system. For instance, in this work, our goal is to improve the equalization performance of an equalizer using a neural network. Within the scope of this project, the following channel models: linear

phase, squared, tanh, and Watterson are used in our simulations when evaluating our proposed techniques. Neural network equalizer has proven its effectiveness in equalizing the signal under the influence of ISI. However, a common assumption in the literature is that the condition of the channel is static or does not change over time. In reality, this assumption may not hold. The condition of a channel can be affected by the weather, temperature, time of the day, etc… [15]. The variation of channel conditions can negatively affect the performance of a neural network equalizer because of how it functions. Before a neural network model can be deployed for any practical application, it must be trained with a large sum of data for it to work effectively [27]. After the training is completed, the weight and biases of a neural network structure are saved, and later on, are used upon deployment. The problem arises when a neural network equalizer is trained on a particular channel condition, but the channel condition changes during deployment which leads to poor equalization performance. The solution to this issue is instead of using one neural network model to be an equalizer, we use a system of neural network equalizers. The idea is to train multiple neural network models with different channel conditions. Once the training of all equalizers in the system is completed, the system of neural network equalizers will be able to account for any changes that the channel may exert.

The second part of this project is to implement a neural network model to perform channel equalization on a High Frequency (HF) channel. The HF channel is typically operated under the center frequency in the range of 3MHz to 30MHz [28]. The most popular channel model for the HF communication channel is the Watterson model, which is named after Watterson Micheal who came up with the model. The Watterson model is comprised of two important features that every realistic channel will consist of [29]. The first feature of the Watterson model is the tapped delay line, which is the mathematical representation for multipath signaling. Depending on geographical location, there can be multiple paths that the signal can travel from the transmitter to the receiver. As a result, when a symbol is sent from the transmitter, there can be multiple copies of that symbol at the receiver at a different time delay [15]. The second feature of the Watterson model is the ability to account for Doppler spread or frequency fading, an effect that is typically associated

with mobility of either transmitter or receiver [30]. One of our goals is to be able to equalize the signal when transmitting over the Watterson channel model. For this work, we take advantage of a built-in Watterson model in MATLAB, which provides us with different channel characteristics based on the channel parameter that the user can choose.

1.2 LITERATURE REVIEW AND RELATED WORKS

1.2.1 Neural Network for Channel Equalization

There have been many efforts in the literature proposing the use of an artificial neural network to perform channel equalization. These efforts have claimed for an improvement in channel equalization performance by comparing to a conventional equalization technique such as zero-forcing equalizer, minimum squared error equalizer, linear equalizer, decision feedback equalizer (DFE), and decision direct equalizer (DDE) [1], [2], [3], [4]. To simulate inter-symbol interferences (ISI) caused by the communication channel, various channel models have been chosen for this work, and a neural network equalizer has been evaluated with these channels. The most commonly used channel in the literature is an FIR filter, which has been used in [1], [2], [5], [6], [4], etc... In addition to an FIR filter channel model, a Rayleigh fading channel has also been used in the simulations of [7], and a digital satellite channel was set up in [8]. To fully investigate the effectiveness of using an artificial neural network equalizer, different neural network architectures were proposed in these efforts. The majority of the works choose to use a feed-forward neural network or multilayer perceptron (MLP) neural network for their proposed model, and several studies have also evaluated the advantage of using more complex neural network architecture such as convolution and recurrent neural network. Particularly, [4] proposed the use of a feed-forward neural network in conjunction with a super-exponential iterative algorithm. Their results showed an improvement in convergent rate and a reduction in mean square error. [5] utilized a convolutional neural network to perform channel equalization on an FIR filter channel. The result [5] has obtained showed that their proposed equalizer can out-perform an MMSE equalizer and a support vector machine equalizer. [2] took an interesting approach by cascading a multilayer perceptron

(MLP) neural network and a radical basic function (RBF) on an FIR filter channel. Their simulations indicated that their proposed model can achieve a lower bit error rate (BER) than a linear, MLP, and RBF equalizer. [7] attempted to use a feed-forward neural network to equalize the signal transmitted over the Rayleigh fading channel. Their results indicated that at the signal-to-noise ratio (SNR) higher than 6 dB, their proposed neural network equalizer is capable of removing channel distortion. [8] used a complex bi-linear recurrent neural network to equalize a signal transmitted over a digital satellite link channel. Their results are compared with a DFE equalizer, which they have shown a better channel equalization performance. [9] proposed a feed-forward neural network equalizer for a frequency selective non-linear multiple-input multiple-output (MIMO) channel. Their Lavenberg-Marquardt algorithm showed good channel equalization performance with lower computational complexity and faster convergence than other algorithms used in the literature.

1.2.2 Epsilon Greedy Algorithm for Neural Network Tuning

when creating a neural network, a process called "hyperparameter tuning" is a necessary step to achieve the peak performance of a neural network for different applications. Tuning a neural network refers to adjusting the value of different hyper-parameter of a neural network such as the number of nodes in each layer, the activation function, the loss function, and the optimizer. The tuning process is typically done by hand-pick from an educational guess of what might be the best hyperparameter options for a particular neural network application. Literature has shown that reinforcement learning can be used to tune a neural network model. To the best of our knowledge, while we have seen works where reinforcement learning techniques are used to tune neural networks for various applications, we have not seen any works using any reinforcement learning algorithm to adjust neural networks for channel equalization. In this section, we will present some examples of using reinforcement learning to tune neural networks for different applications that are most related to our work. For instance, [10] presented an asynchronous reinforcement learning algorithm that finds an optimal network configuration by automatically adjusting parameters for a given problem. Their results showed that

asynchronous reinforcement learning can converge on an optimal solution for the MNIST data set. [11] propose a new reinforcement learning technique for a convolutional neural network to detect fault classification. Their results show that their proposed model can outperform traditional deep learning and machine learning methods. [12], for example, discussed how reinforcement learning is applied to tune a convolutional neural network (CNN) for the detection of global manipulation. Their auto-designed networks can outperform the state-of-the-art manually designed networks in the literature. Additionally, in [13] and [14], a reinforcement learning algorithm is used to tune the structure of adaptive equalizers. In both efforts, the proposed cognitive equalizer tuned with reinforcement learning can provide a better equalization performance.

1.3NEURAL NETWORK EQUALIZER OVERVIEW

An equalizer is a piece of hardware or software at the receiving end of a communication system, which is designed to remove the negative channel effect on a transmitted signal [31]. The conventional techniques of channel equalization include DFE, DD, zero forcing, and linear equalizer, etc.…These types of equalizers have proven their effectiveness in the ability to reconstruct the distorted signal mostly in a linear channel. However, a realistic channel always exerts nonlinear characteristics, therefore the conventional method mentioned above may not be very effective. Past research regarding neural network has provided us with the understanding that neural network is capable of solving complex, and non-linear problem [32]. Thus, many efforts have tried to improve the equalization performance of conventional equalizer structures such as the DFE, DDE, zero forcing equalizer, linear equalizer with a neural network model. Before a neural network equalizer can be deployed to use in practical applications, it has to be trained. At the receiver, after the signal has been transmitted over the channel, it is distorted due to inter-symbol interferences and additive white gaussian noise (AWGN). The corrupted signal is passed thru the input layer of the NN equalizer, and during the training phase, the NN equalizer has access to the original data before it was sent in order for the network to learn. By the method called backpropagation during training, the neural network finds the optimal weight and

biases to minimize the loss, which is the difference between the predicted data and the true data [33].

Figure 3 is the flow chart of how the data is processed thru a communication system with a neural network equalizer. When data arrives at the receiver, it comes as a packet contains a sequence of symbols. In our work, we choose to use a sliding window mechanism, which is a digital signal processing technique [34], is designed to capture a small amount of data within the packet received, and we load this window of data into the input layer of the neural network. The sliding window will continue to slide along the data creating a two-dimensional array with the number of rows is the number of symbols in the packet, and the number of columns is the size of the sliding window which is also a design/hyperparameter of a neural network equalizer.

1.4 NEURAL NETWORK EQUALIZER CREATE WITH KARAS IN TENSORFLOW PLATFORM

Tensorflow is a free open-source software library for machine learning that is developed and distributed by the Google laboratory. Google's toolkit is the most popular machine learning tool that is designed with minimum effort from the user and maximum efficiency. Google Tensorflow can be used with various tasks related to machine learning but has a particular focus on the performance of a deep neural network [35]. Tensorflow is written in three different programming languages: Python, C++, and CUDA, and python is the most popular language among the developer's community. The mathematical operation of TensorFlow is based on the flow of data in which google called a tensor [36]. Since it's initially released in 2015, the platform has grown dramatically and became a major player in the field of machine learning and neural network. The follow-up releases include different versions of Tensorflow that can facilitate implementation on majors computing platforms such as Linux, macOS, Windows, and Android machines. The release of TensorFlow version 2.0 has made implementing and using the library extremely easy and intuitive. Implementing a layer of a neural network is as easy as one single line of code, which contributes to its popularity [37].

The work in this thesis focuses on experimentation of different neural network architectures to come up with a model that yields the highest performance when equalizing a signal sent over a wireless communication channel. Through backpropagation, a neural network can be trained to undertake the task of a channel equalizer. Fine-tuning a neural network is a necessary procedure to achieve its peak performance. The tuning process of a neural network involves experimenting with various hyperparameter options of a neural network that is provided by Karas under the Tensorflow platform. The hyperparameters that developers can experiment with include the number of hidden layers, the number of nodes within a layer, the activation function that governs each node, the optimization method, and the loss function. Each of these parameters or the combination of parameters can yield substantially different results. The motivation for this work is to be able to further improve the use of a neural network to achieve a performance gain when comparing to the conventional equalization techniques such as DFE, DD, LSM, etc.…

1.5 THESIS STRUCTURE

The detailed structure of this work starts with the implementation of a neural network equalizer based on the basic multi-layer perceptron neural network architecture. The channel models used for this implementation are a linear phase channel and High Frequency (HF) channel. The transmitted signal passes thru the channel model, and additive white gaussian noise (AWGN) is added at the output of the channel to simulate various noises that can occur within a communication system including thermal noise [17]. A convolutional neural network equalizer is also investigated, and the equalization performance is evaluated. We then can discuss and analyze the results obtained by comparing the equalization performance of an MLP neural network equalizer and a CNN neural network equalizer.

The next section in this thesis is the proposal of a system of neural network equalizers. In this section, simulations with various channel models including linear

phase, flat fading, squared, cubic, tanh, and HF channels are set up. The results obtained aim to show the novelty of the system of neural network equalizers being able to remove inter-symbol interferences (ISI) while the single neural network equalizer struggles to equalize the signal transmitting over the channel that varies over time.

The next section dedicates to another technique to improve the use of a neural network equalizer, which is hyper-parameter tuning. Particularly, in this section, a reinforcement learning algorithm used to tune different attributes of a neural network equalizer is presented and tested. The assumed channel for these simulations is the HF channel at different channel conditions. The results aim to show an improvement in the equalization performance by using reinforcement learning to tune neural network equalizer compare to a fixed structure neural network equalizer (i.e. by fixed we mean that the neural network can learn the weights but cannot change the value of its hyper-parameters).

The last section is the overall conclusion for all of the work contributed to this thesis. A summary of each section in this work is presented as well as the conclusion based on the results obtained in the simulations. Finally, we suggest several future directions that could further improve the results from this thesis.

CHAPTER 2
NEURAL NETWORK EQUALIZER

2.1 NN EQUALIZER USING MULTILAYERS PERCEPTRON ARCHITECTURE

2.1.1 Simulation Set up

To demonstrate how a neural network equalizer can be used to equalize transmitted signals, a communication system is simulated according to Figure 3. There are two parts to this experiment. The first part of the experiment is when an FIR filter is used as the channel model, since it is a commonly used model in research [5], and the second part is when high frequency (HF) channel is used since it is a more realistic channel, which has many applications in long-distance wireless communication. The Watterson Model [15] is chosen to simulate the High Frequency (HF) channel in this work, as it has been used in many HF-related research efforts. Specifically, the built-in Watterson model available in MATLAB is used for this project. Table 1 provides details on the HF channel conditions used in this work based on their respective time delays and Doppler spreads.

Channel parameter	Good	moderate	poor
Delay spread (ms)	0.1	0.5	1
Doppler Spread (Hz)	0.5	1	2

Table 1: Channel model parameters from ITU recommendation F.520-2 [19].

Random bits are generated from a random bit generator. The bits are then fed into a custom modulation function for the desired modulation scheme, which, in this our case, is BPSK. In order for the data to be processed by the neural network equalizer, the incoming packet of a one-dimensional array is converted into a two-dimensional array. Equations (1) and (2) represent the incoming data to the receiver, and equations (3) and (4) show how the data is being converted into a two-

dimensional array. Equations (2) and (4) represent complex data when HF is the assumed channel. Observe that "*r*" is the real component and "*i*" is the imaginary component of the symbol. The conversion method is how [7] have used in their work, and Figure 1 shows the input layer of their neural network equalizer trained with the complex data.

X_receive=[x0,x1,x2,x3….xn]. (1)

X_receive=[x0r+xoi,x1r+x1i,x2r+x2i,x3r+x3i….xnr+xni]. (2)

X_receive= [[x0,x1,x2,x3,x4] (3)
[x1,x2,x3,x4,x5]
[x2,x3,x4,x5,x6]
….
[xn, 0 , 0 , 0 ,0]].

X_receive= [[x0r+x0i,x1r+x1i,x2r+x2i,x3r+x3i,x4r+x4i] (4)
[x1r+x1i,x2r+x2i,x3r+x3i,x4r+x4i,x5r+x5i]
[x2r+x2i,x3r+x3i,x4r+x4i,x5r+x5i,x6r+x6i]
….
[xnr+xni, 0 , 0 , 0 ,0]].

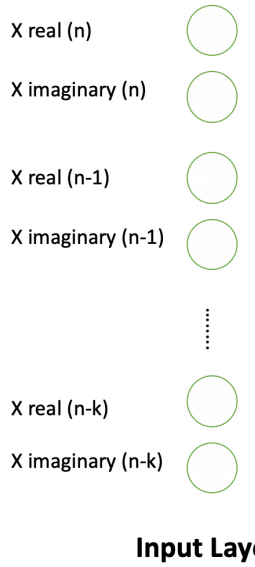

Figure 1: data structure for the input layer of an MLP neural network equalizer [7] has used for their simulation

The above conversion technique is called a sliding window, which is a common digital signal processing technique [34]. Zeros are added to the last few windows to avoid using noises bits from the ISI of the channel. Each sliding window or row in the two-dimensional array is loaded into the input layer to the neural network equalizer, and the first bit of that window is chosen to be the label for the neural network equalizer during training. For instance, if there are 5 bits in each window, then the first window consists of [x0,x1,x2,x3,x4], and the k^{th} window consists of [x(k),x(k+1),x(k+2),x(k+3),x(k+4)]. The label used for the first window is x0, and the label used for the k^{th} window is x(k).

For this experiment, $5*10^5$ bits were generated, and 80% of the bits were used to train the neural network equalizer. The modulation scheme was BPSK, which is represented by 1 and-1 on the constellation map [38]. The sliding window is sized to 10 symbols, and an FIR filter channel model is used for the first part of this experiment whose transfer function is the following:

$$\mathbf{H(z)= 0.3482 + 0.8704\ z^{-1} + 0.3482\ z^{-2}} \quad (5)$$

For the second part of this experiment, HF is the assumed channel. The neural network equalizer used for this experiment is constructed with MLP architecture, and the following is the detail of its structure:

- The input layer consists of 10 nodes, and note that the number of nodes in the input layer has to be the same as the size of the sliding window.
- There are two hidden layers. The first hidden layer has 100 nodes with an activation function "*relu*", and the second hidden layer has 100 nodes with an activation function "*tanh*".
- The output layer has two nodes represent two classes with an activation function "*softmax*".

Figure 2 provides the summary of the Multilayer Perceptron neural network structure created in Karas, which is the high-level API of TensorFlow.

```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 10)                110
_________________________________________________________________
dense_1 (Dense)              (None, 100)               1100
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 202
=================================================================
Total params: 1,412
Trainable params: 1,412
Non-trainable params: 0
```

Figure 2: MLP neural network architecture used for this work.

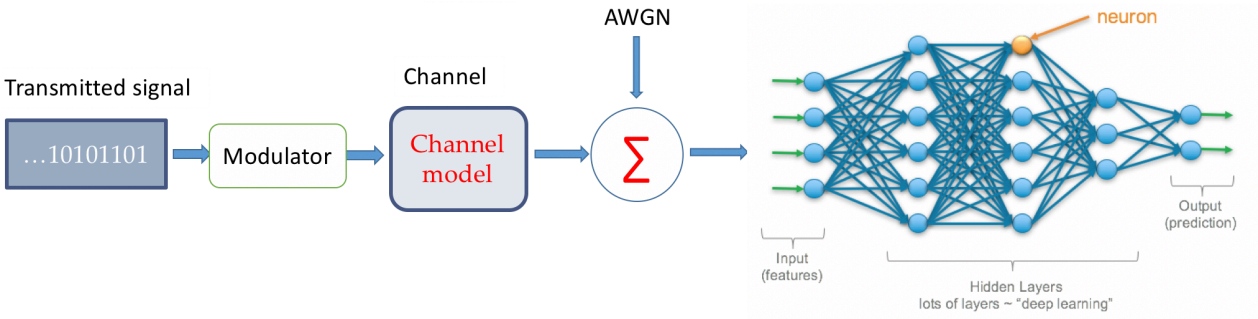

Figure 3: Communication System

2.1.2 Result and Discussion

Part A: Using FIR Filter as a Channel Model

To show that our neural network equalizer can remove the channel distortion, we calculate the bit error rate (BER) on the testing sequence and plot it against signal to noise ratio (SNR). The SNR range is chosen from 0 dB to 16 dB, which is the common range used to evaluate equalizer performance in the literature. Figure 4 shows BER vs SNR, where BER is presented in a logarithmic scale for better visualization. The result from Figure 4 indicates that our proposed neural network equalizer can invert the negative effect from the communication channel. Figure 4 also shows that as the SNR increases, the BER decreases.

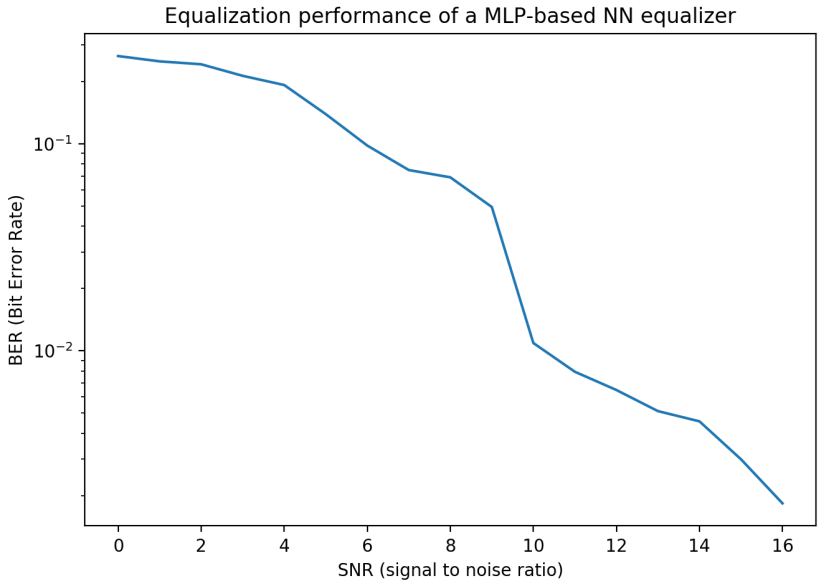

Figure 4: MLP neural network equalizer on FIR filter channel.

Part B: Using HF Channel

For this part of the experiment, we use our neural network equalizer to remove ISI from the HF channel at three distinct channel conditions listed in Table 1. Figures 5-7 indicate that ISI can be removed in the HF channel. However, HF is known to be a difficult channel to equalizer due to its multipath and doppler spread which typically cause severe ISI. The SNR range chosen for this experiment is from 0 dB to 20 dB. Observe that even with the good HF channel condition, which is shown in Figure 5, the equalization performance is worse than that of the FIR filter channel in Figure 4 at the same SNR value. Figures 5-7 also suggest that as the channel condition becomes more severe, it is generally harder for an equalizer to perform well. To also compare the equalization performance of a NN equalizer with other technique for HF channel in the literature, we compare the results in Figure 5 and Figure 8, which is the simulation results that [43] obtained by using an adaptive decision feedback equalizer. By comparing Figure 5 with the 0.5Hz fading rate or doppler spread curve from Figure 8, we can see that the NN equalizer outperforms the adaptive decision feedback equalizer at the same HF channel condition.

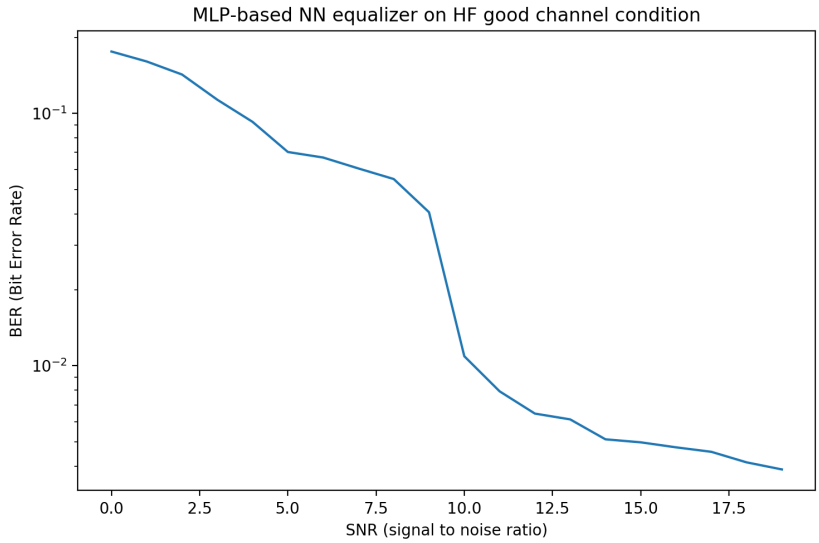

Figure 5: MLP neural network equalizer on HF good channel condition

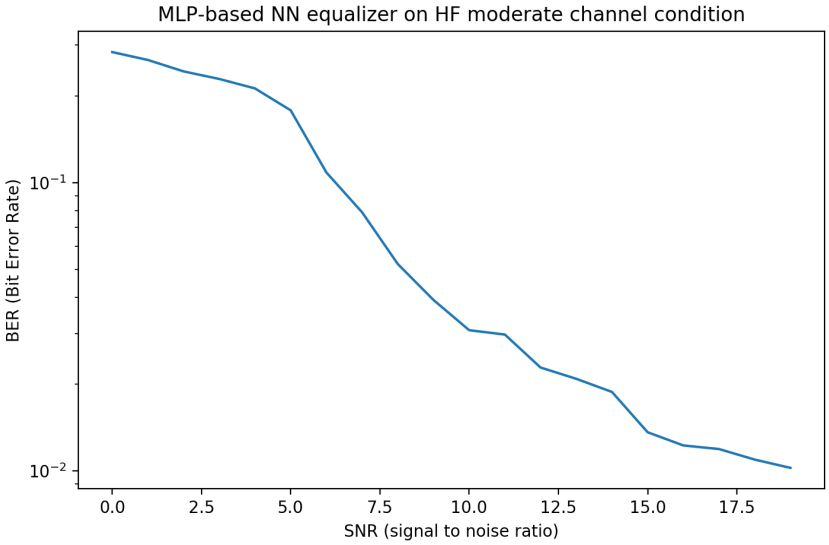

Figure 6: MLP neural network equalizer on HF moderate condition

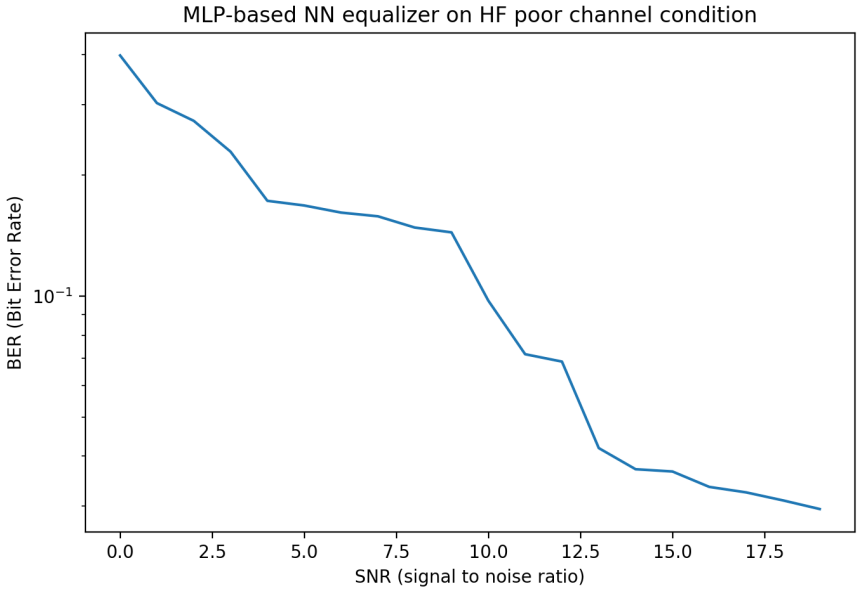

Figure 7: MLP neural network equalizer on HF poor channel condition

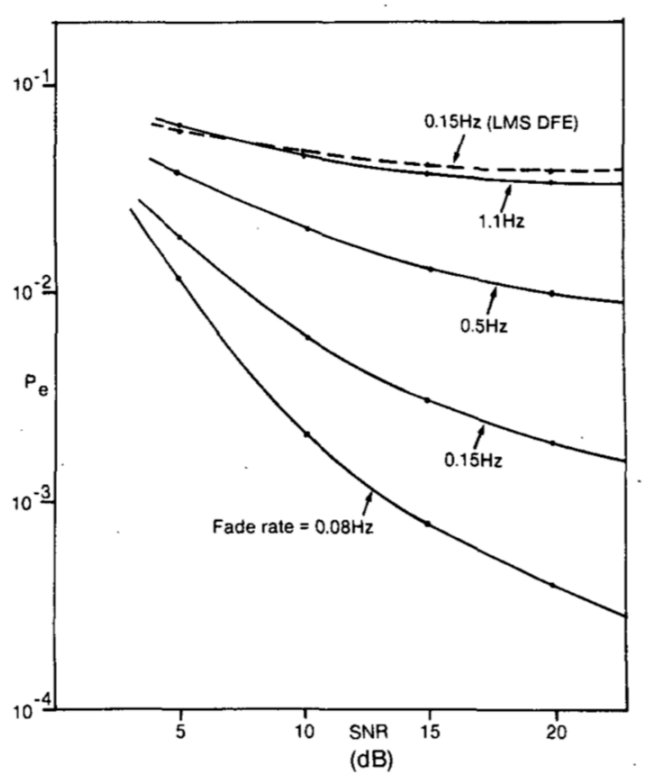

Figure 8: Error Performance vs SNR of an adaptive DFE from [43].

2.2 NN EQUALIZER USING CONVOLUTIONAL ARCHITECTURE

2.2.1 Convolutional Neural Network Architecture

A convolutional neural network is a specific neural network architecture in which data is being processed throughout the neural network in a grid-like arrangement to extract important features of the data [39]. One advantage of a convolutional neural network (CNN) over a multilayers perceptron neural network is the advanced feature extraction capability of two-dimensional data such as images. The given name convolutional neural network is based on the fact that instead of using matrix multiplication in MLP neural network, a convolution operation is used. A CNN model works by applying a filter to the input data, and the output of the filter is usually the features that are critical to a particular application [16]. After filtering, the subsequent layer, which is called max pooling, is used to further refines the most relevant features. The last layer of the CNN architecture is similar to one in the MLP architecture which is the classification layer where a prediction is made about the label of the input data. Figure 9 provides a general structure of a CNN neural network.

Typically, a CNN architecture is more complex than an MLP architecture. In this part of our work, we investigate whether a more complicated architecture such as CNN can yield an equalization performance gain over the feed-forward NN or MLP neural network architecture.

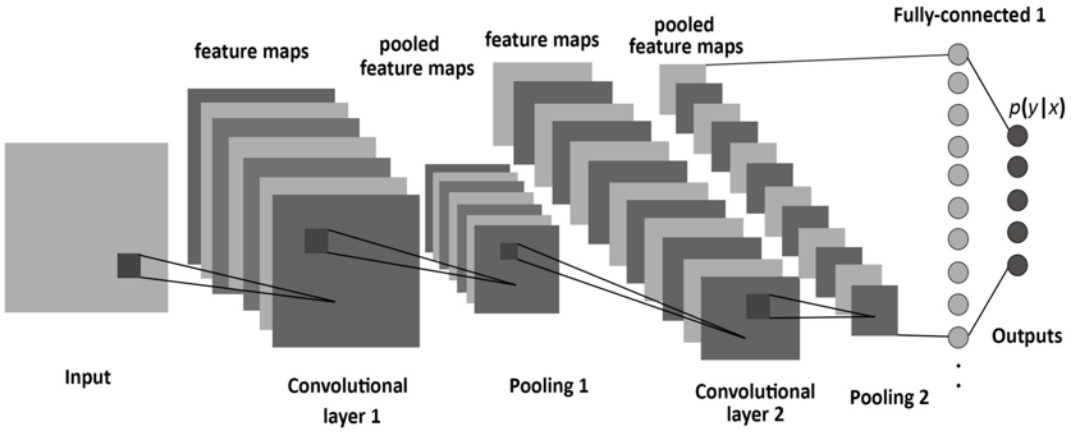

Figure 9: general structure of a convolution neural network

2.2.2 How CNN is Used for Wireless Channel Equalization

As mentioned in the previous section, a CNN is designed to effectively handle two-dimensional data. For this work, a data reconfiguration is necessary before it can be loaded into the input layer of the CNN network. Particularly, a packet of data is transmitted from the transmitter to the receiver, and at the receiver, the packet arrives in the form of a one-dimensional array. A sliding window is designed to create every input frame to the neural network, and at each frame, data is reconfigured from one dimensional to a two-dimensional array. Figure 10 provides the details of the data reconfiguration operation, which is similar to how [16] has used in their work.

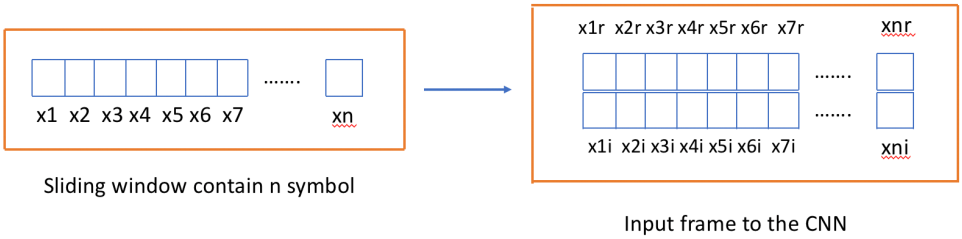

Figure 10: data restructure for the convolution neural network

Each input frame to the convolutional neural network is given a label used to train the neural network equalizer. Particularly, similar to how we train the MLP-based neural network equalizer, the first bit in the sliding window is used as the label to train the network.

2.2.3 Simulation Set up

The motive to evaluate the CNN architecture for the neural network equalizer is to observe whether a CNN architecture can yield better performance over the MLP architecture. Thus, we have set up the two experiments with the same channels used in the previous section which are the FIR filter and the HF channel at three different channel conditions listed in Table 1. Similar to the communication set up from the previous section, random bits are generated and feed into the modulation function to output a BPSK modulated signal. After the signal was sent over the channel, at the receiver, data is processed similarly as the MLP neural

network equalizer. An additional step to reconfigure the data into the form that can be loaded into the CNN network is necessary which is demonstrated in Figure 10. Equation (5) and Table 1 provide the transfer function of the channel and the HF channel condition of the FIR filter and HF channel respectively. Figure 10 is a detailed summary of the CNN structure used in this experiment.

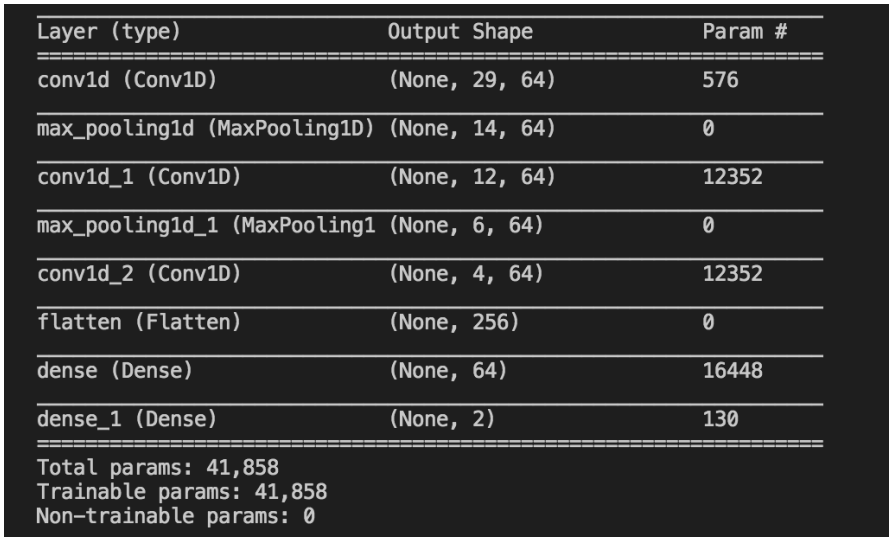

Figure 11: CNN structure used for this work.

2.2.4 Results and Discussion

Part A: FIR filter as the channel model, comparison between MLP and CNN equalizer

To show that a CNN architecture NN equalizer is superior to the MLP architecture NN equalizer, we send the signal over the same channel and compare the equalization result from both CNN and MLP architecture equalizers. The SNR range is chosen from 0dB to 16dB, and the BER is used as the performance measurement. The results indicate that under the FIR filter channel, the CNN architecture NN equalizer yields a noticeable performance gain at every SNR within the evaluation range. However, the CNN performance gain comes at the cost of computational complexity. Particularly it takes approximately three times longer to train a neural network equalizer using CNN architecture than using MLP architecture with the same amount of data. Figure 11 shows the results of the equalization performance comparison between a CNN and MLP architecture neural network equalizer on the FIR filter channel.

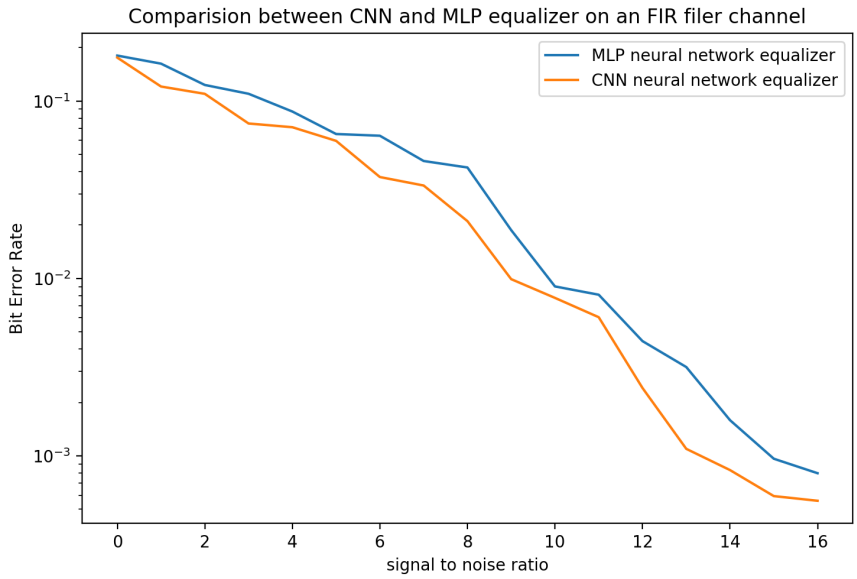

Figure 12: comparison between CNN and MLP equalizer on an FIR filer channel.

Part B: HF channel, comparison between MLP and CNN equalizer

For this experiment, we simulate a communication system with an HF channel. The SNR range is from 0 dB to 20 dB, and we compare the equalization performance between a CNN architecture NN equalizer and an MLP architecture NN equalizer. Figures 13-15 suggest that the general trend still holds for both equalizers that as the HF channel conditions get worsen in terms of time delay spread and Doppler spread, the equalization performance degrades. In every channel condition, a CNN equalizer can outperform an MLP equalizer at each SNR within the SNR range used in this work.

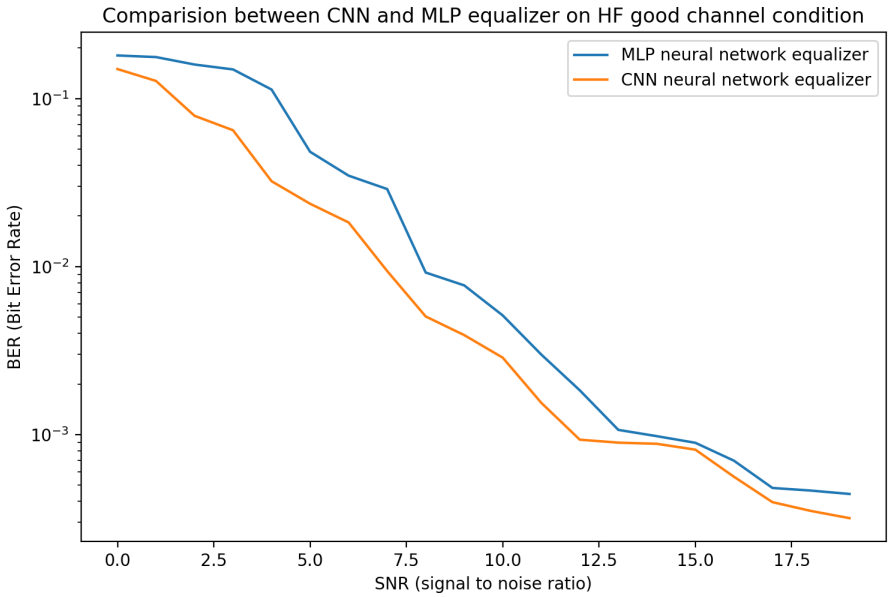

Figure 13: compare between CNN and MLP equalizer on HF good channel condition.

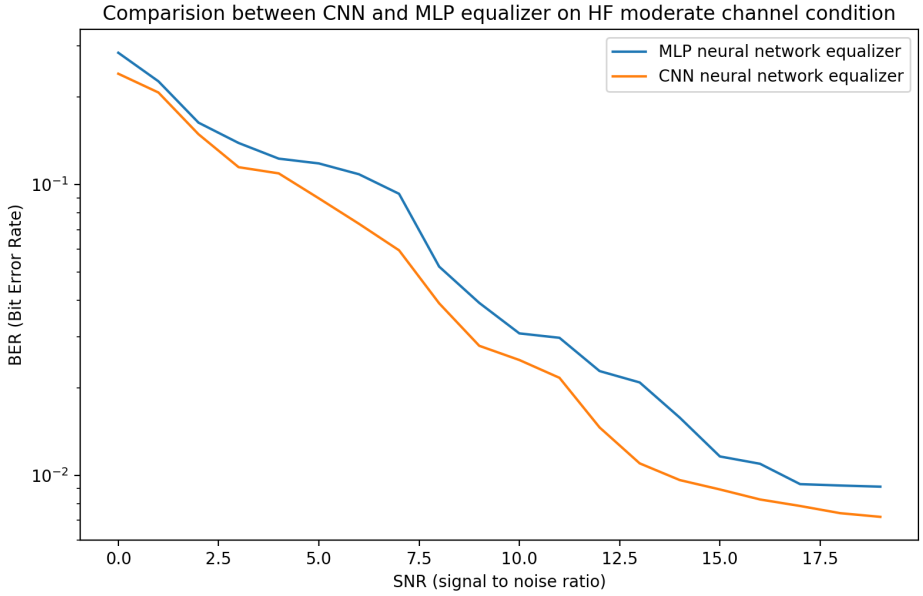

Figure 14: compare between CNN and MLP equalizer on HF moderate channel condition.

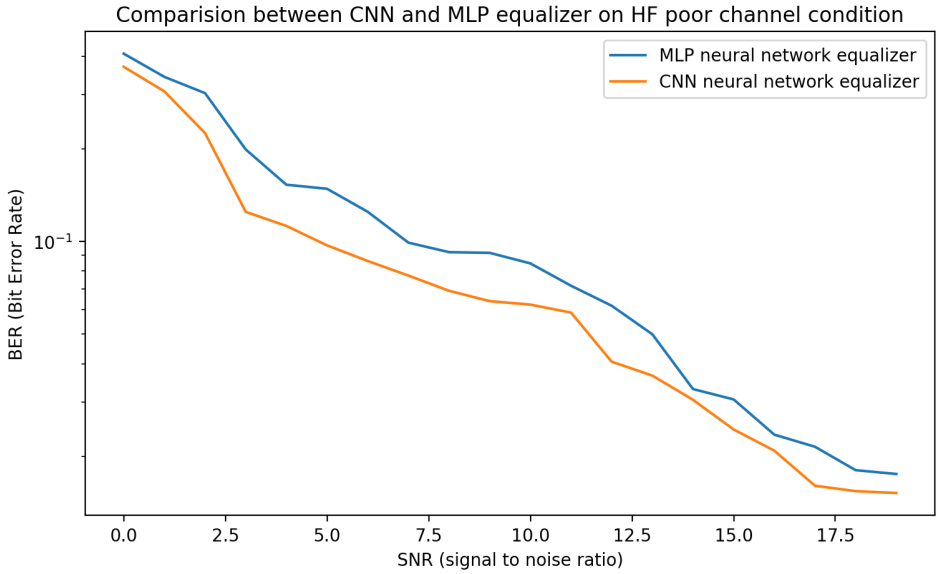

Figure 15: compare between CNN and MLP equalizer on HF poor channel condition.

2.3 CHAPTER SUMMARY

In this chapter, a case study of neural network equalization was conducted. We have constructed two different neural network equalizers based on two different neural network architectures, which are multilayer perceptron (MLP) and convolutional neural network (CNN). Both equalizers are tested on the FIR filter channel, and the HF channel at three different channel conditions listed in Table 1. Two conclusions can be drawn from the results obtained. The first conclusion is that the convolutional neural network equalizer yields slightly better performance than the multilayer perceptron neural network equalizer at the cost of more complexity and complicated data structure. The second conclusion is that, in general, a channel with a more severe ISI will be more challenging for an equalizer to remove the distortion from the transmitted signal.

CHAPTER 3
SYSTEM OF NEURAL NETWORK EQUALIZERS

3.1 WHY USING SYSTEM OF NEURAL NETWORK EQUALIZERS

Neural networks have been proven in the literature to be an effective tool for channel equalization, especially on the non-linear channel due to their ability to solve complex problems. However, a neural network equalizer is limited to a specific set of channel characteristics or channel conditions, it is therefore not effective on the channel whose condition can change over time such as the High Frequency (HF) channel. Particularly, the conditions of the HF channel may change due to various factors including temperature, time of the day, weather condition, etc.… [18]. To overcome the limitation of a neural network equalizer on the channel whose characteristic may change, a neural network equalizer can be retrained with a new set of data that can be collected when the channel condition changes. However, having to retrain a neural network equalizer every time the channel condition changes is both inconvenient and computationally expensive. To address this issue, we propose a system of neural network equalizers designed to equalizer the signal transmitted over the channel whose conditions may change over time.

3.2 HOW THE SYSTEM OF NEURAL NETWORK EQUALIZERS WORK

The proposed system of neural network equalizers consists of multiple pre-trained neural network equalizers. Each neural network equalizer is essentially a module that connects to the system of neural networks. As indicated by the block diagram in Figure 16, data transmission is divided into packets, and each packet of data contains information bits, which is the actual/desired data, and preamble bits. Preamble bits are known at the receiver, which provides a means for the equalizer to learn about the channel [17]. When a packet arrives at the receiver, the preamble bits are passed into the input layer of each neural network equalizer in the system. The process continues with all of the pre-trained neural network equalizers performing equalization on the preamble bits. Since the preamble bits are known

bits at the receiver, bit error rates (BER) can therefore be calculated. The outputs from all of the NN equalizers are fed into a decision-maker block that chooses the equalizer yielding the lowest BER of the preamble bit for the packet. The decision can be calculated as follows

$$\boldsymbol{D = \min(Y_1, Y_2, Y_3, \ldots Y_N)} \quad , \tag{6}$$

where D is the decision variable, and Y1, Y2, Y3,…, Y_N is the BER of the neural network equalizers on the preamble bits from 1 to N. After the decision-maker block reaches a decision, the information bits or actual data in that packet are then loaded into the input layer of the chosen NN equalizer to be equalized. The process is repeated for every packet until all of the data is equalized.

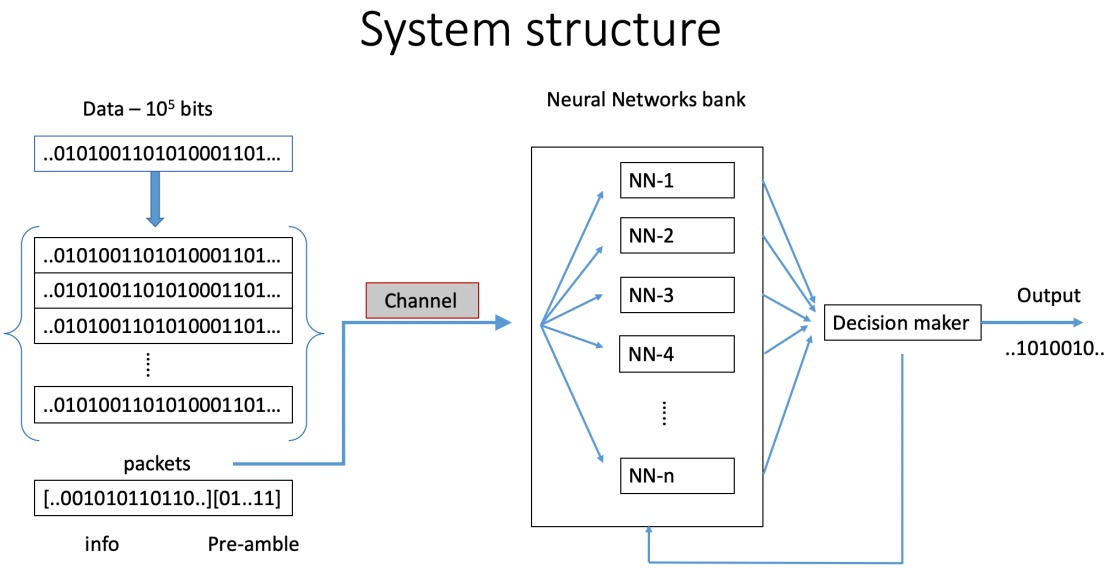

Figure 16: System of neural network equalizers

3.3 APPLYING SYSTEM OF NEURAL NETWORK EQUALIZERS ON SELECTED CHANNEL

3.3.1 Simulation Set up

An experiment was conducted to evaluate the performance of the system of the neural network equalizers compared to an individual neural network equalizer. For simulations, we use the baseband BPSK (i.e., +1 and -1) as the modulation scheme. Our neural network models were built using the TensorFlow 2.0 platform in the Python programming language. The data was transmitted over a channel that varying with time. The transmitted data was divided into packets, and after every 20 packets, the channel changes structure. The experiment assumes a fixed SNR of 12 dB, and the BER, which is used as a measure of equalizer performance, is calculated at each packet. The channel model sequentially changes in the following order: linear phase, squared, flat Rayleigh fading, cubic, and tanh. The linear phase channel model was used in [5] for their simulation, and the impulse response of the channel can be represented by Equation (5).

A total of $5*10^5$ bits were used in each trial of this part of our work, and we average the results over 100 trials. The data is divided into 100 packets, and each packet contains 5000 bits. The system consists of 5 NN equalizers, and each equalizer is modeled with Multi-layer Perceptron NN architecture, which is detailed as follows:

- The input layer has 10 nodes.
- The output layer has two nodes with the activation function "*softmax*".
- There are three hidden layers total:
    - the first hidden layer has 40 nodes with a "relu" activation function
    - the second hidden layer has 40 nodes with a "tanh" activation function
    - the last hidden layer has 10 nodes also with a "tanh" activation function

- The optimizer is “Adamax”, which is a method for stochastic optimization with an adaptive learning rate.
- The loss function is “sparse-categorical-cross-entropy”, which is a type of cross-entropy loss without the need for one-hot encoding.

There are 10^4 bits of the data used to train each neural network equalizer in the system. The training cycle repeats 10 times for the given training data. L2 regularization is implemented to prevent overfitting when training each neural network equalizer. The idea behind L2 regularization is to try to reduce model overfitting by keeping the magnitudes of the weight value small [20]. After a neural network equalizer was trained with a specific channel condition, the optimized weights and biases were saved and labeled as a pre-trained model, which later on can be loaded into the system of neural network equalizers.

3.3.2 Results and Discussion

There are two separate simulations for this part of the project. The first simulation involves evaluating the performance of a single neural network equalizer as the channel structure varies. The second simulation employs our system of neural network equalizers on the same varying channel. These experiments are conducted to simulate various arbitrary channels and the equalization performance of each model. A total of 100 packets are sent, where the channel changes every 20 packets. The channel variations occur in the following order:

- The first 20 packets will be sent over the Linear Phase channel
- The next 20 packets will be sent over the Squared channel
- The next 20 packets after that will be sent over the Flat Fading channel
- The next 20 packets after that will be sent over the Cubic channel
- The last 20 packets will be sent over the Tanh channel.

Figures 17-21 show the BER calculated for each packet of a single neural network equalizer when it is trained on one of the aforementioned channels as it is exposed to the channel changes in the order described previously. The results

indicate that a single neural network equalizer will not be able to equalize the signals transmitted over a channel whose structure varies over time. Figures 17-21 also suggest that the performance of the equalizers on a particular packet of data depends on the channel model the signal is sent over. Based on the calculated BER of each packet, the equalizer performance fluctuates because it can only equalize the signal when the signal has been transmitted on the specific channel that the equalizer was trained on. It is clear that when the neural network equalizer is exposed to the channel model it was trained on, its performance is significantly better. In Figure 19, for example, the equalizer was trained on a flat Rayleigh fading channel model, and the signal was transmitted through a flat fading channel between the 40th and 60th packets. The bit error rate is significantly less during this portion of the simulation compared to when the signal is transmitted over the other channel models. The same trend can be observed with neural network equalizers that have been trained on the other channel models as shown in Figures 17-21. These fluctuations verify that a single neural network equalizer can only be effective when it is exposed to the channel it was trained on.

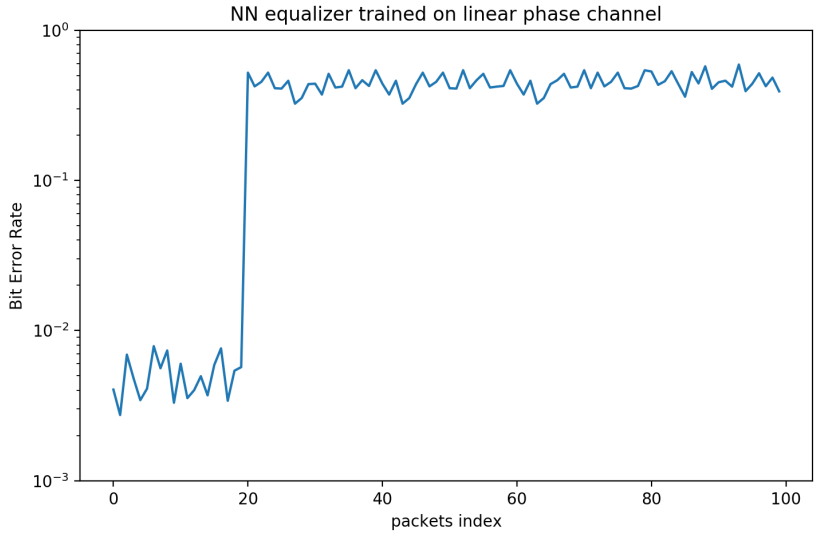

Figure 17: NN equalizer trained on linear phase channel and test on a channel that varies over time.

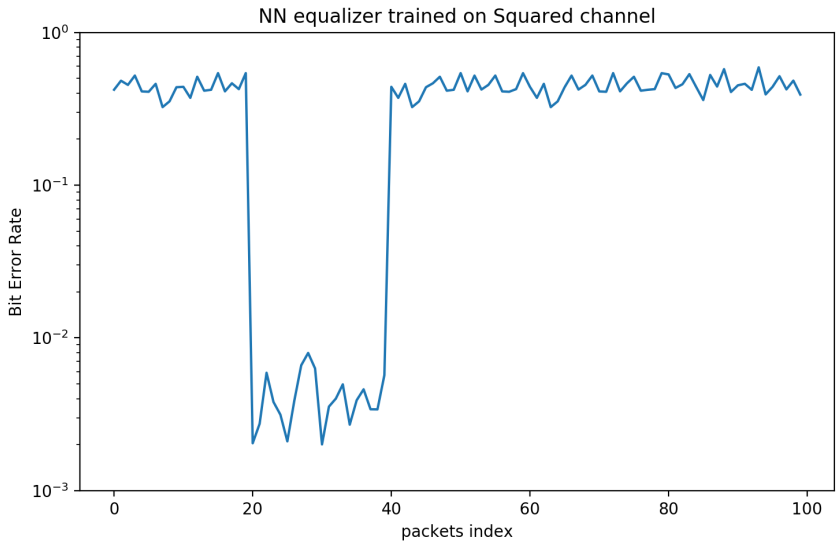

Figure 18: NN equalizer trained on squared channel and test on a channel that varies over time.

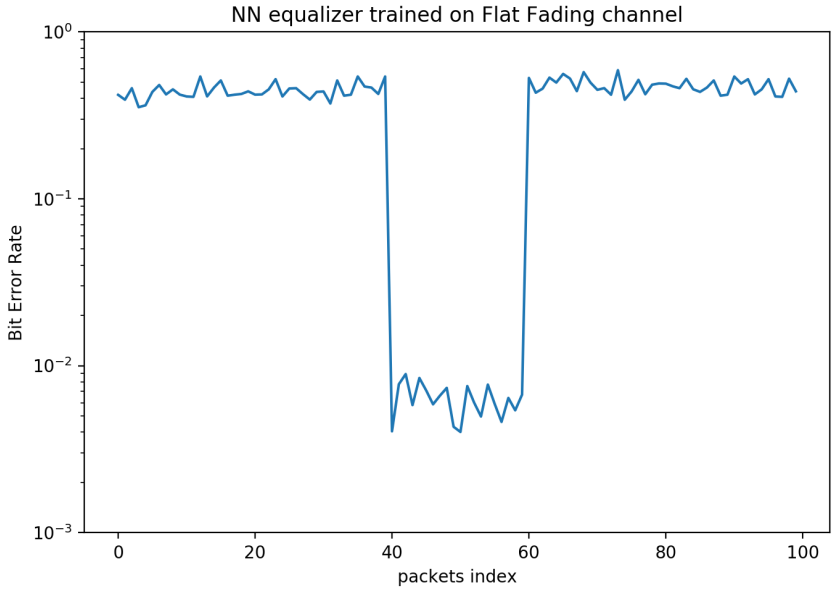

Figure 19: NN equalizer trained on flat fading channel and test on a channel that varies over time.

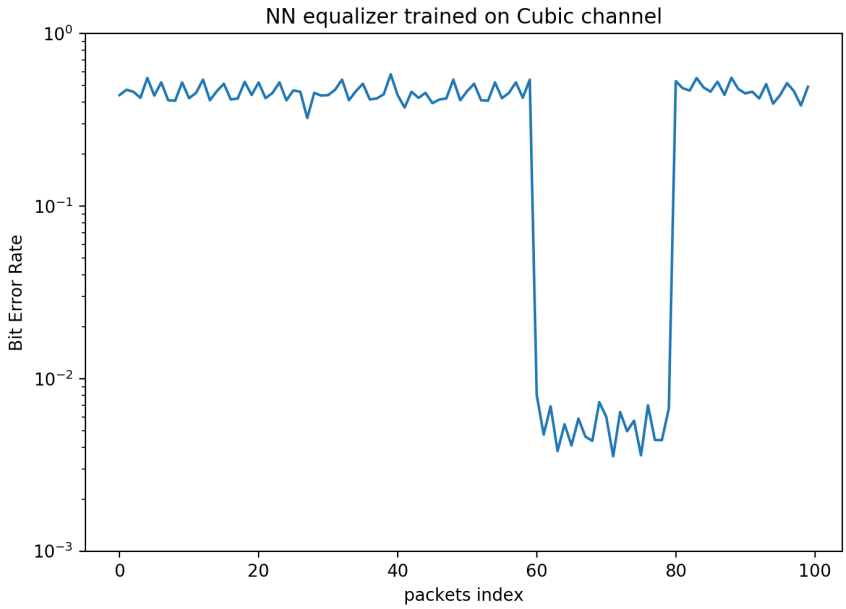

Figure 20: NN equalizer trained on Cubic channel and test on a channel that varies over time.

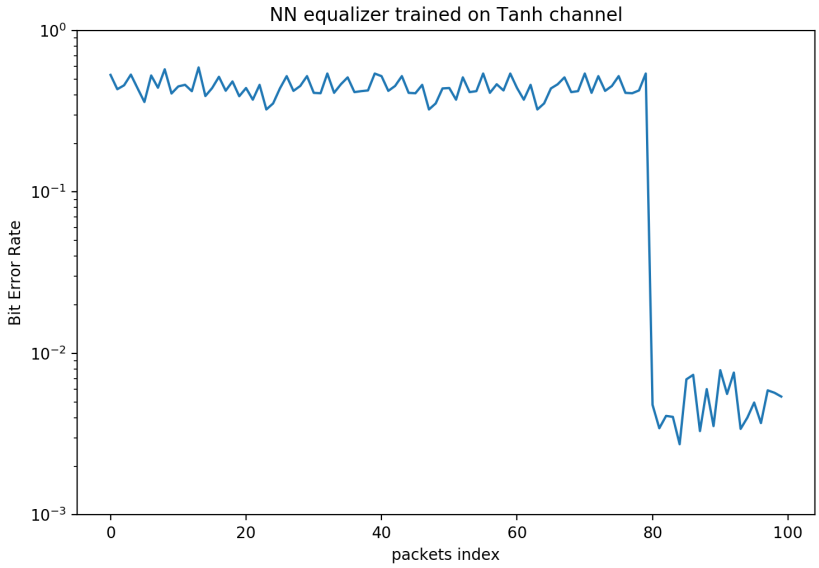

Figure 21: NN equalizer trained on Tanh channel and test on a channel that varies over time.

The second simulation involves using the system of neural network equalizers to equalize signals transmitted over the same sequence of channel models that were used in the previous experiment. The motivation behind using the system of neural network equalizers is to choose a neural network equalizer that yields the lowest BER when equalizing the preamble bits to ensure the peak performance for all of the transmitted packets. In this section, the system of neural network equalizers consists of five single neural network equalizers, each trained on one of the five channel models described previously (i.e., linear, square, flat Rayleigh fading, cubic, and tanh). Figure 21 indicates that the channel variations do not affect the performance of the system of neural network equalizers, and shows that a single neural network equalizer trained on linear phase channel only works when the channel is linear phase.

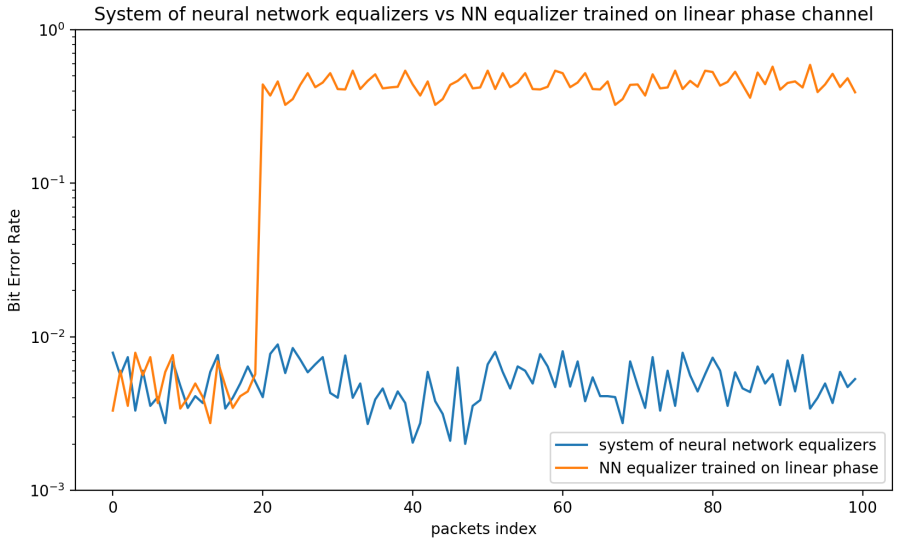

Figure 22: system of neural network equalizers vs NN equalizer trained on linear phase channel, both test on channel that varies over time.

3.4 APPLYING SYSTEM OF NEURAL NETWORK EQUALIZERS ON HF CHANNEL

3.4.1 Simulation Set up

The first portion of simulations discussed in this section is conducted by sending the signal over the selected channel models. We then proceed to deploy the system of neural network equalizers to the High Frequency (HF) channel to examine its performance on a more realistic channel. In this part of the experiment, when a High Frequency (HF) channel is assumed, and we use a built-in HF channel model in MATLAB for simulation. The HF channel is chosen due to its global interest and propensity for having frequent and time-varying properties. The HF channel transmits signals with a carrier frequency in the range of 3 to 30 MHz [18]. It supports long-distance communications by enabling a signal to bounce multiple times off the ionosphere [15]. To conduct this portion of the experiments, the Watterson model is used to simulate the HF communication channel. The Watterson channel model is a popular channel model used to simulate HF channel which was proposed by C. Watterson and his colleagues [15]. The ISI that the HF channel causes are typically in the form of multipath, and Doppler spread [15]. Different channel conditions of the Watterson model are categorized by the multipath differential time delay (often called delay spread) and fading rate (often called Doppler spread). Table 1 provides the detail of the different HF channel conditions used in this work [19].

3.4.2 Results and Discussion

Figures 5-7 show the results of a single neural network equalizer in different HF channel conditions that we obtained, which are detailed in Chapter 2. The results are plotted to show BER vs SNR and are averaged over 100 trials. The purpose of these simulations is to show that the single neural network equalizers are capable of equalizing the signal in different HF channel conditions so that they can be included in the system of neural network equalizers. We trained three separate neural network models on three channel conditions specified in Table 1. We then validate the performance of the neural network equalizer at each channel condition. Figures 5-7 also indicate that the neural network equalizer trained on the

good channel obtains the best performance, which is expected because the good channel has the least degradation because of the minimum ISI caused by the channel. The equalization performance is exacerbated as the channel degradation becomes more severe. For example, at 15 dB, the bit error rates are $3*10^{-3}$, $2.6*10^{-2}$, $7.9*10^{-2}$ for the good, moderate, and poor channel conditions respectively.

After having analyzed the single neural network equalizer with different HF channel conditions, we proceeded to use the system of neural network equalizers to mitigate the effect of varying HF channel. The three neural network equalizers that were used in simulations to generate results in Figures 5-7 constitute the system of neural network equalizers in this part of the experiment, as they were shown to work in their respective HF channel conditions. The validation test is similar to the first experiment with the selected channel models from section 3.3. There are 100 trials are conducted for this part of the experiment, and for each trial, 60 packets are sent over the HF channel with the SNR kept fixed at 12dB. After every 20 packets, the channel condition changes. Two tests are conducted to analyze the performance of the system. The first test changes the channel conditions from good to moderate to poor, the results are shown in Figures 23-26. Figures 23-25 show the performance of single neural network equalizers on the same channel variation. These plots indicate that when the condition of the channel is the same as the channel condition the neural network equalizer has been trained on, it's able to equalize the transmitted signal. Conversely, when the condition of the channel was different from the channel condition the neural network equalizer was trained on, the equalizer struggles to remove the channel impairments. The second test changes the channel conditions from poor to moderate to good, and the results are plotted in Figures 27-30. The purpose of the second test is to ensure that the results do not depend on how the channel conditions change. Figures 27-29 convey the same idea as Figures 23-25 which indicate that the single neural network equalizer is not capable of equalizing channels whose conditions change over time.

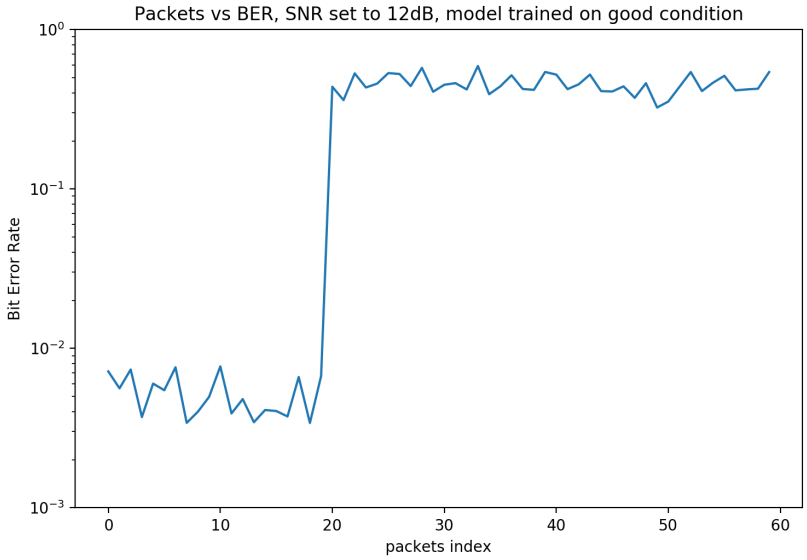

Figure 23: NN equalizer train on HF good channel condition and test on HF with channel condition change from good to moderate to poor.

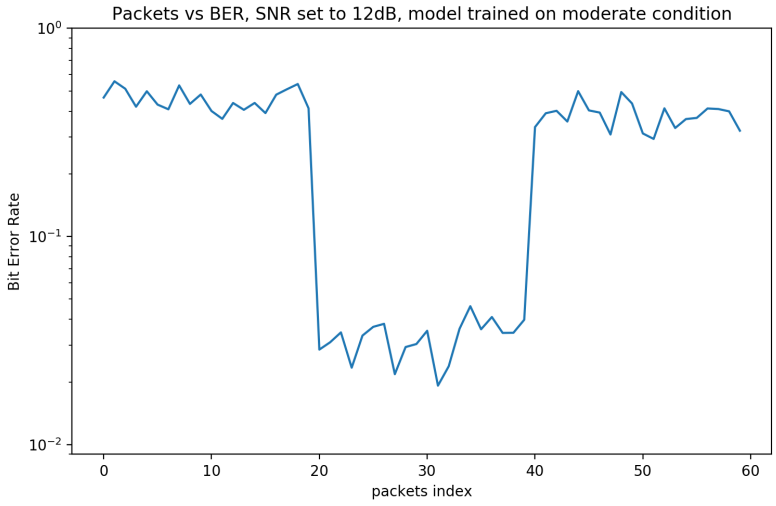

Figure 24: NN equalizer train on HF moderate channel. condition and test on HF with channel condition change from good to moderate to poor.

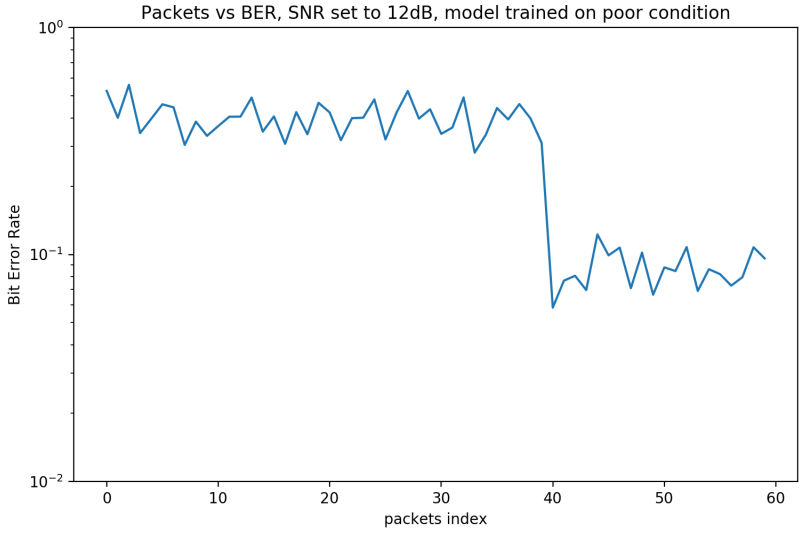

Figure 25: NN equalizer train on HF poor channel condition and test on HF with channel condition change from good to moderate to poor.

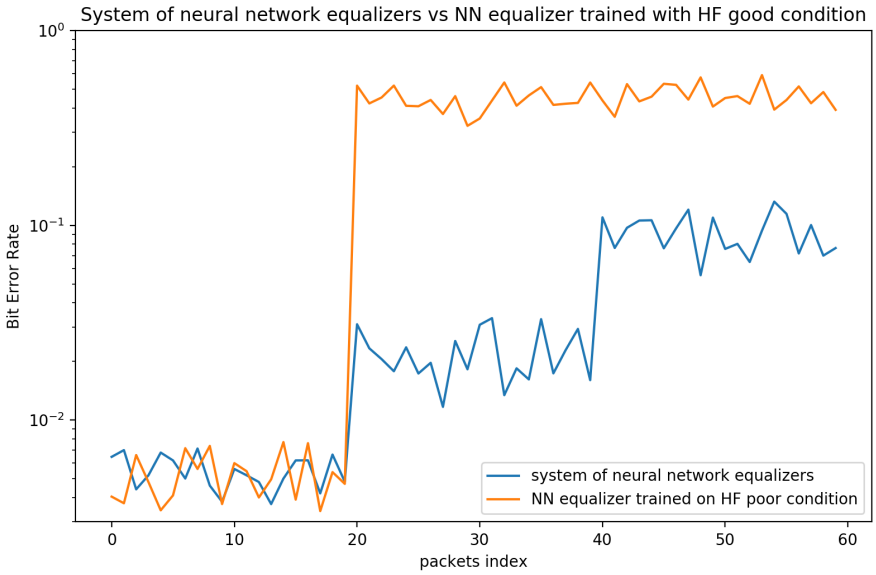

Figure 26: system of NN equalizers vs NN equalizer train on HF good channel, both test on HF channel that conditions vary from good to moderate to poor

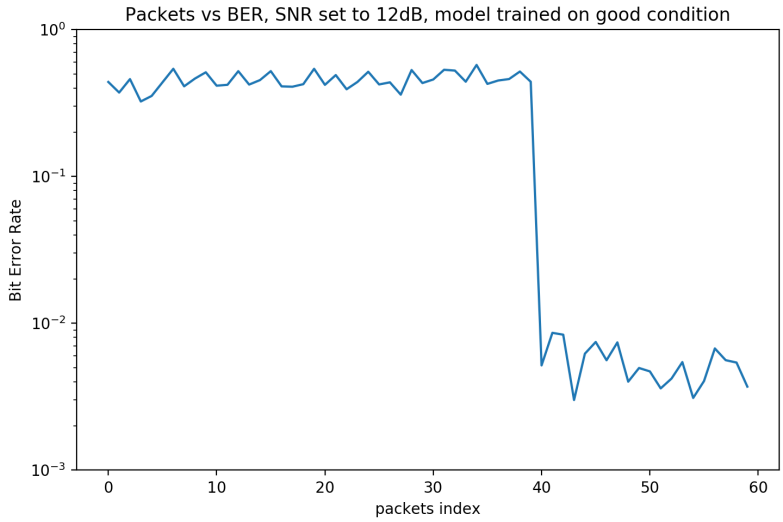

Figure 27: NN equalizer train on HF good condition and test on HF with condition change from poor to moderate to good.

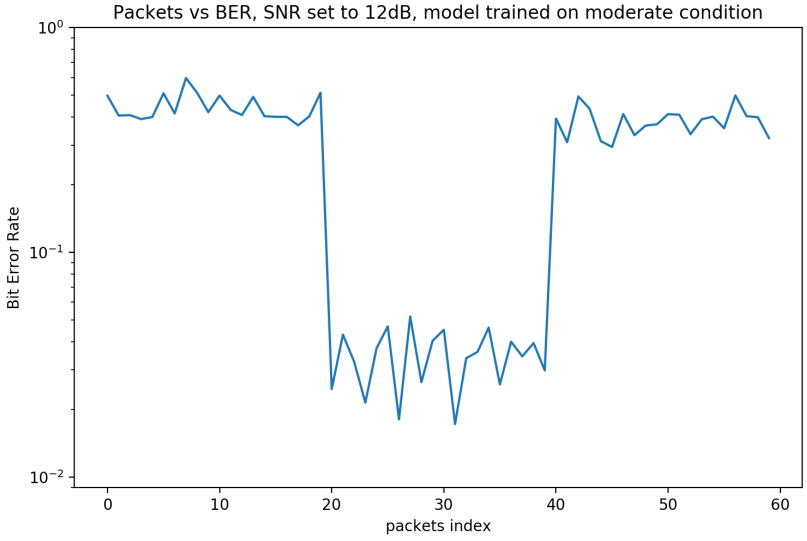

Figure 28: NN equalizer train on HF moderate condition and test on HF with condition change from poor to moderate to good.

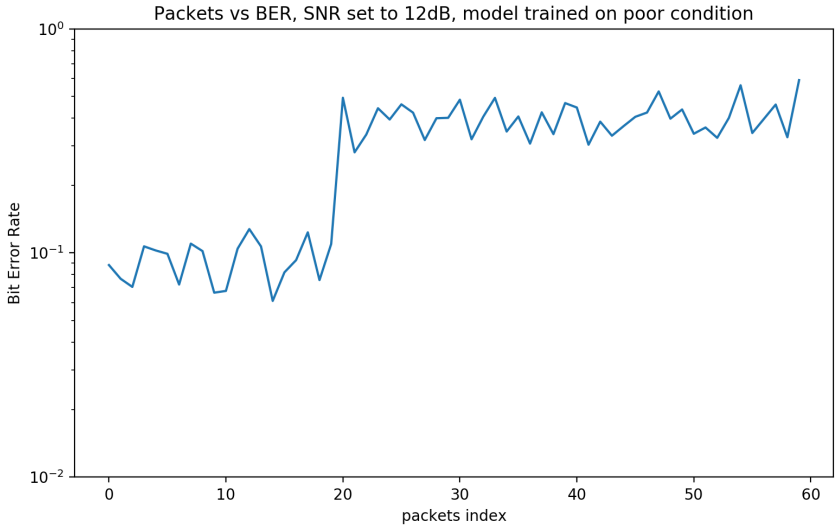

Figure 29: NN equalizer train on HF poor condition and test on HF with condition change from poor to moderate to good

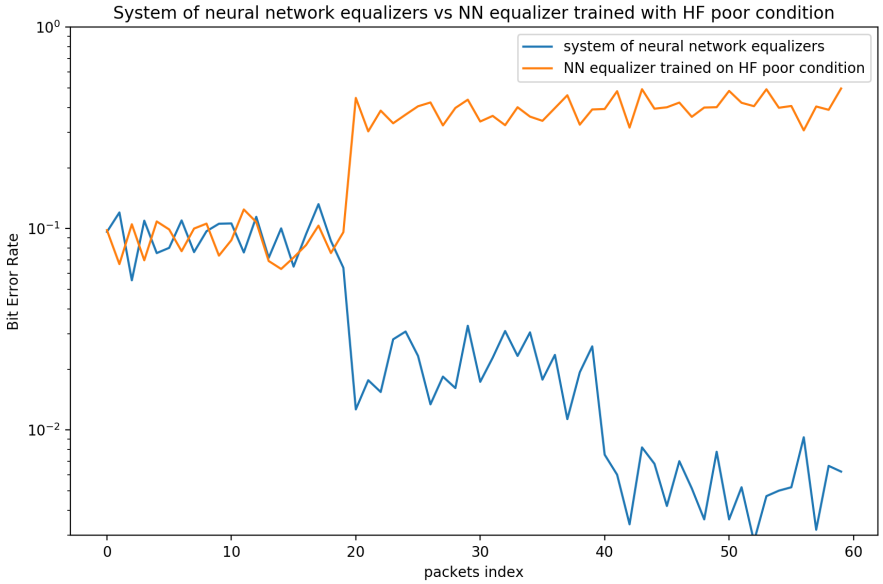

Figure 30: system of NN equalizers vs NN equalizer train on HF poor channel, both test on HF channel that conditions vary from poor to moderate to good

The plots from these Figures 23-25 and 27-29 indicate that the NN equalizer can only equalize the portion of the data sent over the same channel condition as it was trained on. In Figure 23 for example, on the packet indices from 0 to 20th, the calculated bit error rate shows that the equalizer was able to equalize the signal. However, between the 40th and 60th packets, the bit error rate suggests that the equalizer could not equalize the signal. In contrast to the single neural network equalizer, Figures 26 and 30 are the results when our proposed system of neural network equalizers was employed. These two Figures compare the performance of a single neural network equalizer to our system of neural network equalizers transmitting over the same varying HF channel conditions. The results demonstrate that when the channel condition changes, the system of Neural Network equalizers can select the best pre-trained Neural Network equalizer to mitigate the ISI under varying channel conditions.

3.5 CHAPTER SUMMARY

In this chapter, we propose a system of neural network equalizers to overcome
the limitation of a single neural network equalizer on the channel whose condition may change over time. A detailed explanation of how the system works as well as how the data is processed throughout the system is provided. To evaluate the performance of our proposed model, we set up two different experiments comparing the equalization performance of a single neural network equalizer to our system of neural network equalizer. For the first experiment, we selected five channel models which are linear phase, squared, flat fading, cubic, and tanh. For the second experiment, we chose to use three different HF channel conditions which are listed in table 1. From the results we were able to obtain, a conclusion can be drawn that our proposed system of neural network equalizers is capable of equalizing the distorted signal that has been transmitted on a channel whose condition changes over time while a single neural network equalizer struggles to remove channel impairments under the same channel variation.

CHAPTER 4

TUNING NEURAL NETWORK EQUALIZER USING EPSILON GREEDY ALGORITHM

4.1 INTRODUCTION TO EPSILON GREEDY ALGORITHM

4.1.1 Reinforcement Learning

Reinforcement learning is a subset of machine learning where agents (i.e., users) take actions in an environment in order to maximize the notion of cumulative reward [40]. A reinforcement learning model does not need a large collection of data because its target label is unknown [21]. On the other hand, a supervised learning model, which is another popular subset of machine learning, requires a large dataset with labels to be effective, and depending on the application, obtaining a large dataset with labels can be challenging. In reinforcement learning, the network that transforms input data to output action is called "policy network", the simplest way to train a policy network is the method called "policy gradient". The learning process started with a completely random network, then a random output action is produced. The random output action is then fed back to the learning engine to produce the next input frame, and the reward is given to each action taken to complete the learning loop [41]. If the learning agent chooses to explore, then a random option is taken from the list of available options to observe the reward from the environment. On the other hand, when the agent chooses to exploit, an option the agent has previously explored that has yielded the highest reward will be chosen. The objective of reinforcement learning is to find the policy that yields the highest reward by using rewards/penalties given to the agent based on its choice of action(s) [40].

4.1.2 Epsilon Greedy Algorithm

The Epsilon greedy algorithm is a simple reinforcement learning technique that has proven to be effective in many applications. Epsilon is the probability of the learning agent exploring rather than exploiting [42]. At each time step, to determine whether the agent will explore or exploit, a random number between 0 and 1 is generated. If the random number is greater than epsilon, the agent will choose to exploit instead of explore. The objective of the algorithm, like all other reinforcement learning algorithms, is to learn the actions that maximize the reward the agent can obtain. Figure 31 shows the pseudocode of the epsilon greedy algorithm.

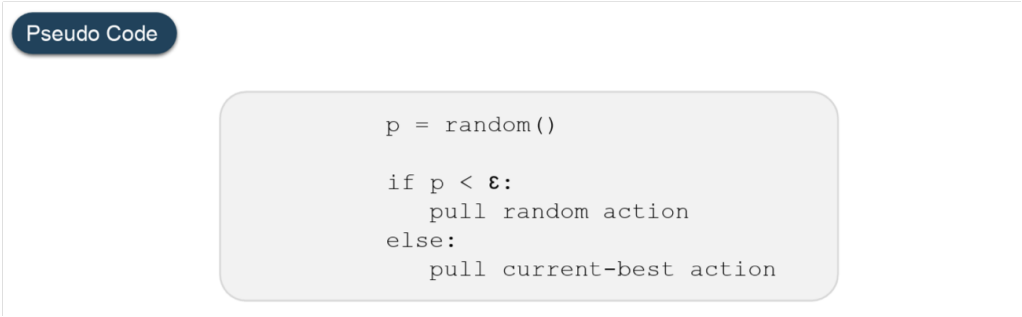

Figure 31: Pseudocode of epsilon greedy strategy.

4.2 NN EQUALIZER TUNED WITH EPSILON GREEDY ALGORITHM

The epsilon greedy algorithm, as explained in the previous section, attempts to balance between exploration and exploitation. When the agent is exploring, the algorithm randomly picks a new option consisting of different neural network hyper-parameters; such as the number of nodes in a layer, the activation function, and the optimizer [12]. When the agent is exploiting, the agent will pick an option with the highest reward among all of the options that the agent has previously explored. As mentioned in section 4.1.2, the rewards help the agent learn which

option to choose. The actual reward given to the agent depends on the application. In this work, the ultimate objective is to minimize the bit error rate (BER), therefore BER should be used as the reward. However, in a real-time wireless communication system, the true values of the transmitted data are only known during the training phase, not the testing phase. Thus, BER cannot be used as the reward for the epsilon greedy algorithm. In our project, we use a metric called the minimum average probability distance as the reward, which is the running sum of the difference between the probability that 1 or -1 was originally sent to the prediction probability of the symbol that the NN classified as, which outputted by the "*softmax*" function of the NN output layer divided by the total number of symbols. For example, if the NN equalizer predicts 97.5% that the transmitted symbol is -1, then the probability distance is 1-0.975=0.025. The minimum average probability distance concept is similar to the minimum average distance metric used in [13], which is a running average of the minimum distances between the equalized symbols and the symbols of the BPSK constellation divided by the length of the bits sent in the testing sequences.

Within the scope of this project, we investigate the effectiveness of using the epsilon greedy algorithm to tune a feedforward neural network or multilayer-perceptron NN equalizer. The following is the list of available hyperparameters that can be tuned:

- Number of nodes in the input layer
- Number of nodes in the hidden layers
- Activation functions
- Optimizer

At each time step, a training and testing sequence are concatenated and sent through the channel, and the agent can either explore or exploit. When the agent is exploring, a random number of nodes for the input layer and the hidden layers, an activation function for each layer, and an optimizer will be picked from the list of available options [12]. After all of these tunable parameters have been selected by the learning agent, the neural network starts its training phase. When the training

is completed with a chosen set of hyperparameters, the minimum average probability distance is calculated over the testing sequence and used as a reward for the learning agent. For this work, we use an annealing epsilon greedy algorithm, where the value of epsilon decays after every time step resulting in the learning agent likely to choose exploit over explore [14]. The benefit of this strategy is that at the beginning of the experiment, the learning agent is more likely to explore to find an option that yields a high reward, but, over time, the agent is more likely to exploit to take advantage of the option that yields the highest rewards it has already observed during exploration.

4.3 RESULTS AND DISCUSSION

4.3.1 Simulation Set up

For this project, an experiment is conducted to compare the equalization performance of a fixed neural network equalizer, which means the neural network can learn the weights but cannot change the value of its hyper-parameters, and a neural network equalizer tuned with reinforcement learning. The objective of this work is to show the effectiveness of using reinforcement learning to tune a neural network equalizer, by showing that an improvement in equalization performance is obtained when compared with a fixed neural network equalizer. The results obtained are plotted in the form of BER vs SNR, and at each SNR, 50 trials are conducted to get the average result. The High Frequency (HF) band is the assumed channel in this work, due to its popularity for attaining long-distance communications. BPSK is the assumed modulation scheme for our simulation. The Watterson Model from [15] is chosen to simulate the High Frequency (HF) channel, as it has been used in many HF-related research efforts. Specifically, the built-in Watterson model available in MATLAB is used for this project. Table 1 provides the details on the HF channel conditions used in this work based on their respective time delays and Doppler spreads [19].

Our neural network is constructed using TensorFlow, a machine learning platform using Python. We chose to use the annealing epsilon greedy algorithm for our learning agent. The reason for choosing the annealing epsilon greedy is that the

value of epsilon is gradually decreased such that the chances of exploring will decrease significantly as the epsilon value gets close to zero. By having the epsilon value decay after every action taken by the learning agent, we can reduce the probability of yielding low rewards as a consequence of exploring too frequently. Essentially, over time the agent will choose to exploit the options that have yielded high rewards. The method in which epsilon decays in this work is represented in the following equation [22]

$$\boldsymbol{\varepsilon = \frac{\varepsilon_0}{1+n*d}} \quad , \tag{7}$$

where ε_0 is the initial value of epsilon, d is the decaying rate, and n is the time step index which is initialized to zero. For our experiment, we set $\varepsilon_0 = 1$, $d = 0.03$, and the maximum number of time steps per signal to noise ratio (SNR) to 500. By choosing $\varepsilon_0 = 1$, we essentially force the learning agent to explore first. The hyper-parameters of the neural network that the epsilon greedy algorithm has access to includes the number of nodes in each layer except the output layer, the activation functions, and the optimizer. At the input layer, due to the input data being complex, the number of nodes has to be an even number to pass the real and imaginary components of the data separately into our neural network equalizer [7]. The Epsilon Greedy algorithm can vary the number of neurons in the input layer to be an even number between 2 to 100. It can also choose the number of neurons in each hidden layer (even or odd) between 2 to 100. The activation functions available for the epsilon greedy are hyperbolic tangent (*tanh*), sigmoid, rectified linear unit (*relu*), and the exponential linear unit (*elu*). The optimizers available for tuning include *adam*, *adamax*, and Stochastic Gradient Descent (*SGD*). The attributes of the output layer cannot be adjusted by the epsilon greedy algorithm, due to the nature of the problem. Particularly, for our application, there are two nodes in the output layer, representing the points -1 and +1 in the BPSK constellation. These are the two classes that the neural network can classify a received symbol as. Subsequently, the activation function used at the output layer is *softmax*. To compare this work with what is already in the literature, we chose to set the fixed neural network equalizer similar to what [7] has used in their simulation. Particularly, the input layer has 10

nodes, there is one hidden layer with 30 nodes and the activation function is "*tanh*", the output layer has 2 nodes using the *softmax* activation function. *Adamax* is the optimizer used for this model. This fixed neural network equalizer is one of the options available to the epsilon greedy algorithm.

4.3.2 Results and Analysis

For analysis, we chose to directly compare the equalization performance of the tuned Neural Network equalizer to a fixed neural network equalizer whose hyperparameters are hand-picked. Figures 32-34 show the results obtained from the simulations for the Good, Moderate, and Poor HF channel conditions respectively. The plots in these Figures indicate that at any given SNR in the testing range from 0 dB to 20 dB, an epsilon greedy tuned neural network equalizer can outperform a fixed structure neural network equalizer. Note that as the HF channel conditions get worse (Figures 32-34), the fixed NN equalizer and the epsilon greedy tuned NN equalizer are exacerbated. The reason for this performance degradation is because as the HF channel behavior in terms of delay spread and Doppler spread becomes more severe, it is generally harder for an equalizer to perform well.

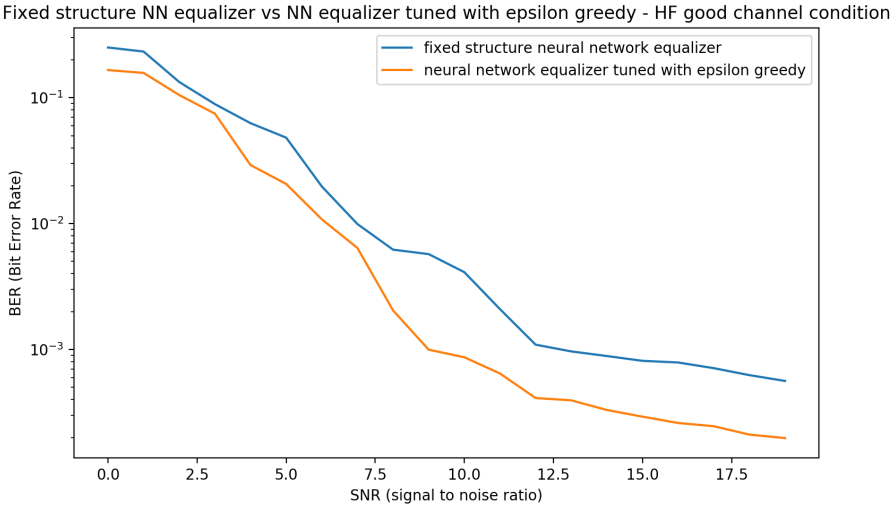

Figure 32: BER vs SNR of Fixed NN equalizer and epsilon greedy tuned NN equalizer on HF good condition.

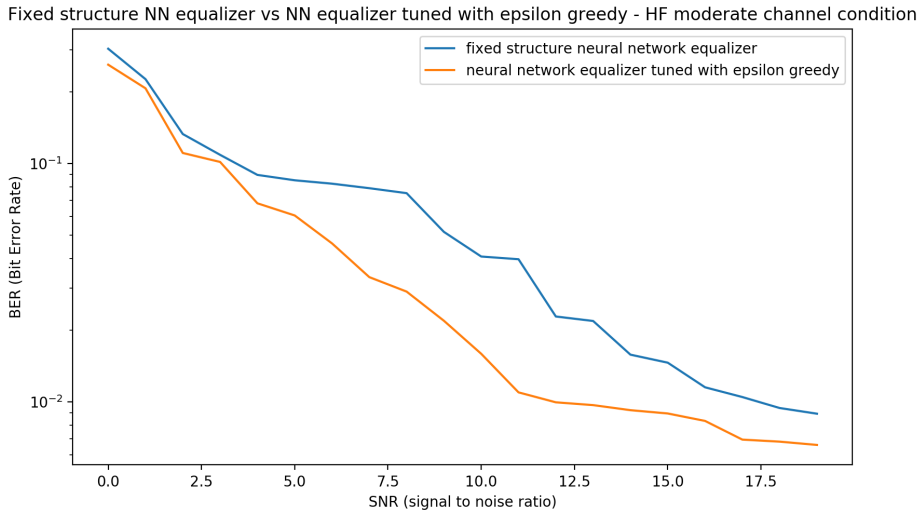

Figure 33: BER vs SNR of Fixed NN equalizer and epsilon greedy tuned NN equalizer on HF moderate condition.

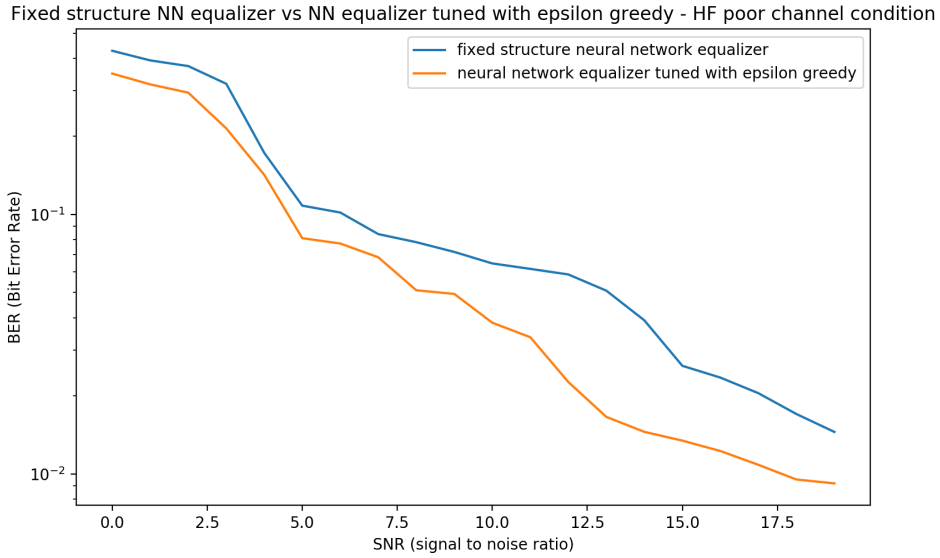

Figure 34: BER vs SNR of Fixed NN equalizer and epsilon greedy tuned NN equalizer on HF poor condition.

4.4 CHAPTER SUMMARY

In this effort, the epsilon greedy algorithm is presented as a method for tuning a neural network equalizer. Our simulation indicates that by using reinforcement learning to tune our neural network equalizer, we were able to

achieve a performance improvement compared to using a fixed neural network equalizer. The motivation for this work is based on the fact that when a neural network is constructed, it always needs to be tuned to achieve good performance. Reinforcement learning has been proven to be an effective method to tune a neural network in other applications, and in our work, our results have shown that reinforcement learning can be used to tune NN equalizers to obtain an improvement in channel equalization performance.

CHAPTER 5
CONCLUSION AND FUTURE WORK

This work has introduced two different techniques to improve the use of neural networks for wireless channel equalization. For the first technique, we propose a system of neural network equalizers, and for the second technique, we investigate the use of reinforcement learning to tune the neural network equalizer. A system of neural network equalizers is shown to significantly improve the equalization performance on the channel that its condition can change over time. To fully show the effectiveness of our proposed system, we conduct two experiments. In the first experiment, a list of channel models including linear phase, flat Rayleigh fading, squared, cubic, and tanh are used to simulate the channel that the conditions change with time. A system of neural network equalizers consisting of five neural network equalizers that have previously been trained with five aforementioned channels is compared with a single neural network equalizer under the same channel variations. For the second experiment, we test our system on the same scenario as the first experiment but with the HF channel, which is a more realistic channel that is widely used in practical wireless communication applications. The channel variation is simulated by changing the condition of the HF channel from good to moderate to poor, and from poor to moderate to good. From both experiments for this part of our work, the results obtained indicate that our system of neural network equalizers is capable of removing ISI from the channel with the condition vary over time whereas the single neural network equalizer struggles to remove the ISI from the signal. In another effort to further improve the effectiveness of utilizing the neural network equalizer, we investigate the use of reinforcement learning to tune different attributes of a neural network equalizer such as the number of nodes in each layer, the activation function, and the optimizer. Particularly, the reinforcement learning technique we choose for our work is called the annealing epsilon greedy algorithm, which is a special version of the epsilon greedy algorithm in which its epsilon value decays over time. HF is once again the assumed channel in this effort due to its popularity in global

communication. With the same set of data, we compare the equalization performance of a fixed neural network equalizer and an epsilon greedy tuned neural network equalizer under three HF channel conditions which are good, moderate, and poor. The results obtained suggest that by using the epsilon greedy to tune the attributes or hyperparameters of the neural network equalizer, better channel equalization can be achieved.

To continue with the subject regarding the deep learning techniques for wireless channel equalization, several future directions are worth investigating. In chapter 2 when we present a neural NN equalizer using MLP architecture and convolutional architecture, we conclude that a convolutional neural network yields a better equalization performance at the small cost of computational complexity. One future direction can be to evaluate the equalization performance of a recurrent neural network equalizer by comparing it with MLP and CNN architecture equalizers. In chapter 3 when we introduce the concept of a system of neural network equalizers. Our system has proven to be an effective tool to equalize the signal sending over a channel that the condition varies with time. However, our system only consists of neural network equalizers using MLP architecture. To further improve the equalization capability, we can add a CNN, or RNN neural network equalizer to the system. In chapter 4 when we present a reinforcement learning technique to tune a neural network equalizer, we only limit our experiment to MLP architecture. An investigation regarding using the same tuning technique in chapter 4 to tune a neural network equalizer with CNN architecture can be a promising future direction to continue this work. Finally, to increase the throughput of an existing communication system, multiple-input multiple-output (MIMO) is the proper direction. With that in mind, to further improve the work in the subject of deep learning for wireless channel equalization, we can apply the system of neural network equalizers and the epsilon greedy tuned neural network equalizer on a MIMO system to observe whether higher throughput can be achieved while maintaining the equalization performance.

REFERENCES

[1]. H. K. Sahoo and B. Mohanty, "Adaptive decision feedback equalizer for SISO communication channel using combined FIR-neural network and fast block LMS algorithm," 2016 IEEE Annual India Conference (INDICON), 2016, pp. 1-5, doi: 10.1109/INDICON.2016.7839048.

[2]. B. Lu and B. L. Evans, "Channel equalization by feedforward neural networks," 1999 IEEE International Symposium on Circuits and Systems (ISCAS), 1999, pp. 587-590 vol.5, doi: 10.1109/ISCAS.1999.777640.

[3]. T. Miyajima and T. Hasegawa, "Performances of decision feedback equalizers using neural networks under frequency selective fading channels," [Proceedings] Singapore ICCS/ISITA `92, 1992, pp. 374-378 vol.1, doi: 10.1109/ICCS.1992.254926.

[4]. M. Gao, Y. Guo, Z. Liu and Y. Zhang, "Feed-Forward Neural Network Blind Equalization Algorithm Based on Super-Exponential Iterative," 2009 International Conference on Intelligent Human-Machine Systems and Cybernetics, 2009, pp. 335-338, doi: 10.1109/IHMSC.2009.92.

[5]. W. Xu, Z. Zhong, Y. Be'ery, X. You and C. Zhang, "Joint Neural Network Equalizer and Decoder," 2018 15th International Symposium on Wireless Communication Systems (ISWCS), 2018, pp. 1-5, doi: 10.1109/ISWCS.2018.8491056.

[6]. K. Burse, R. N. Yadav and S. C. Shrivastava, "Channel Equalization Using Neural Networks: A Review," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 40, no. 3, pp. 352-357, May 2010, doi: 10.1109/TSMCC.2009.2038279.

[7]. A. Elsidig and S. Babiker, "Rayleigh Fading Channel Equalization using Neural Networks," 2018 International Conference on Computer, Control, Electrical,

and Electronics Engineering (ICCCEEE), 2018, pp. 1-5, doi: 10.1109/ICCCEEE.2018.8515895.

[8]. Dong-Chul Park and Tae-Kyun Jung Jeong, "Complex-bilinear recurrent neural network for equalization of a digital satellite channel," in IEEE Transactions on Neural Networks, vol. 13, no. 3, pp. 711-725, May 2002, doi: 10.1109/TNN.2002.1000135.

[9]. O. B. Belkacem, R. Zayani, M. L. Ammari, R. Bouallegue and D. Roviras, "Neural network equalization for frequency selective nonlinear MIMO channels," 2012 IEEE Symposium on Computers and Communications (ISCC), 2012, pp. 000018-000022, doi: 10.1109/ISCC.2012.6249262.

[10]. P. Neary, "Automatic Hyperparameter Tuning in Deep Convolutional Neural Networks Using Asynchronous Reinforcement Learning," 2018 IEEE International Conference on Cognitive Computing (ICCC), 2018, pp. 73-77, doi: 10.1109/ICCC.2018.00017.

[11]. L. Wen, X. Li and L. Gao, "A New Reinforcement Learning based Learning Rate Scheduler for Convolutional Neural Network in Fault Classification," in IEEE Transactions on Industrial Electronics, doi: 10.1109/TIE.2020.3044808.

[12]. Y. Chen, Z. Wang, Z. J. Wang and X. Kang, "Automated Design of Neural Network Architectures With Reinforcement Learning for Detection of Global Manipulations," in IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 5, pp. 997-1011, Aug. 2020, doi: 10.1109/JSTSP.2020.2998401.

[13]. N. Teku and T. Bose, "Cognitive equalization for hf channels", *International Telemetering Conference 2018*, 2018.

[14]. N. Teku, H. Asadi, T. Bose, and M. Marefat, "Fully cognitive transceiverfor High Frequency (HF) applications," in *Open Architecture/OpenBusiness Model Net-Centric Systems and Defense Transformation* 2019, R. Suresh, Ed., vol. 11015, International Society for Opticsand Photonics.SPIE, 2019, pp. 11 – 19. [Online]. Available:https://doi.org/10.1117/12.2515407

[15]. C. Watterson, J. Juroshek and W. Bensema, "Experimental Confirmation of an HF Channel Model," in IEEE Transactions on Communication Technology, vol. 18, no. 6, pp. 792-803, December 1970, doi: 10.1109/TCOM.1970.1090438.

[16]. Y. Li, M. Chen, Y. Yang, M. Zhou and C. Wang, "Convolutional recurrent neural network-based channel equalization: An experimental study," 2017 23rd Asia-Pacific Conference on Communications (APCC), 2017, pp. 1-6, doi: 10.23919/APCC.2017.8304090.

[17]. Lathi, B. P., & Ding, Z. (2019). *Modern digital and analog communication systems* Oxford University Press.

[18]. Wilson, J. M. (2011). *A low power HF communication system* (Unpublished doctoral dissertation).

[19]. ITU, I. (1993). *Use of high frequency ionospheric channel simulators*

[20]. Q. Han, Y. Fan, J. Ge, X. Cui and D. Yu, "Training Ternary Neural Networks By Rectified L2 Regularization," 2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT), 2019, pp. 231-234, doi: 10.1109/ICCASIT48058.2019.8973113.

[21]. W. Wu and M. Liao, "Reinforcement Fuzzy Tree: A Method extracting Rules from Reinforcement Learning Models," 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), 2019, pp. 47-51, doi: 10.1109/ICIS46139.2019.8940165.

[22]. A. Shahidullah, H. Asadi, H. Volos, B. Ryu, and T. Bose, "Cognitiveradio experiment design for the space communications and navigation(scan) testbed," 03 2015.

[23]. W. T. Illingworth, "Beginner's guide to neural networks," in IEEE Aerospace and Electronic Systems Magazine, vol. 4, no. 9, pp. 44-49, Sept. 1989, doi: 10.1109/62.35668.

[24]. M. Zamkotsian, K. P. Peppas, F. Lazarakis and P. G. Cottis, "Layered Offset Hierarchical QAM Modulation for Intersymbol Interference Reduction," in IEEE Communications Letters, vol. 17, no. 11, pp. 2176-2179, November 2013, doi: 10.1109/LCOMM.2013.091213.131697.

[25]. C. Lei, L. Pengpeng, W. Yiwei, L. Zhijun and G. Ou, "Analysis of High-Order IF Wideband Group Delay Equalization for Satellite Receiver," 2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control, 2013, pp. 1317-1322, doi: 10.1109/IMCCC.2013.293.

[26]. J. Wang, A. Al-Kinani, W. Zhang, C. Wang and L. Zhou, "A general channel model for visible light communications in underground mines," in China

Communications, vol. 15, no. 9, pp. 95-105, Sept. 2018, doi: 10.1109/CC.2018.8456455.

[27]. D. Liu, "Open-loop training of recurrent neural networks for nonlinear dynamical system identification," IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222), 2001, pp. 1215-1220 vol.2, doi: 10.1109/IJCNN.2001.939534.

[28]. C. Ning and Y. Hongyi, "A Novel Structure of Signal Demodulation in High Frequency (HF) Channel," 2009 International Conference on Signal Processing Systems, 2009, pp. 107-111, doi: 10.1109/ICSPS.2009.19.

[29]. L. Ming, L. Jianqiang and J. Hua, "Automatic Classification of Modulations in the Flat Fading HF Channel Based on Watterson Model," 2010 Second International Workshop on Education Technology and Computer Science, 2010, pp. 218-221, doi: 10.1109/ETCS.2010.39.

[30]. W. N. Furman and E. Koski, "Standardization of an intermediate duration HF channel variation model," The Institution of Engineering and Technology 11th International Conference on Ionospheric radio Systems and Techniques (IRST 2009), 2009, pp. 1-5, doi: 10.1049/cp.2009.0051.

[31]. M. Q. Le, P. J. Hurst and J. P. Keane, "An adaptive analog noise-predictive decision-feedback equalizer," 2000 Symposium on VLSI Circuits. Digest of Technical Papers (Cat. No.00CH37103), 2000, pp. 216-217, doi: 10.1109/VLSIC.2000.852895.

[32]. P. Henrique and G. Coelho, "Adaptive channel equalization using EKF-CRTRL neural networks," Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290), 2002, pp. 1195-1199 vol.2, doi: 10.1109/IJCNN.2002.1007664.3

[33]. J. Hertz, A. Krogh, B. Lautrup and T. Lehmann, "Nonlinear backpropagation: doing backpropagation without derivatives of the activation function," in IEEE Transactions on Neural Networks, vol. 8, no. 6, pp. 1321-1327, Nov. 1997, doi: 10.1109/72.641455.

[34]. C. -I. Chang, Y. Wang and S. -Y. Chen, "Anomaly Detection Using Causal Sliding Windows," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 8, no. 7, pp. 3260-3270, July 2015, doi: 10.1109/JSTARS.2015.2422996.

[35]. F. Ertam and G. Aydın, "Data classification with deep learning using Tensorflow," 2017 International Conference on Computer Science and Engineering (UBMK), 2017, pp. 755-758, doi: 10.1109/UBMK.2017.8093521.

[36]. D. Demirović, E. Skejić and A. Šerifović–Trbalić, "Performance of Some Image Processing Algorithms in Tensorflow," 2018 25th International

Conference on Systems, Signals and Image Processing (IWSSIP), 2018, pp. 1-4, doi: 10.1109/IWSSIP.2018.8439714.

[37]. *Tutorials : TensorFlow Core*. TensorFlow. (n.d.). https://www.tensorflow.org/tutorials/.

[38]. M. Alhalabi, F. I. El-Nahal and N. Taşpinar, "Comparison of different modulation techniques for optical OFDM Intensity Modulation and Direct Detection IM/DD system," 2019 IEEE 7th Palestinian International Conference on Electrical and Computer Engineering (PICECE), 2019, pp. 1-4, doi: 10.1109/PICECE.2019.8747252.

[39]. McGregor, M. (2021, April 28). *What Is a Convolutional Neural Network? A Beginner's Tutorial for Machine Learning and Deep Learning*. freeCodeCamp.org. https://www.freecodecamp.org/news/convolutional-neural-network-tutorial-for-beginners/#:~:text=A%20convolutional%20neural%20network%20is%20a%20specific%20kind,to%20do%20a%20lot%20of%20pre-processing%20on%20images.

[40]. freeCodeCamp.org. (2020, April 7). *An introduction to Reinforcement Learning*. freeCodeCamp.org. https://www.freecodecamp.org/news/an-introduction-to-reinforcement-learning-4339519de419/.

[41]. Xiao-ting Cui and Xiang-dong Liu, "Fuzzy Neural Control of Satellite Attitude by TD Based Reinforcement Learning," 2006 6th World Congress on Intelligent Control and Automation, 2006, pp. 3983-3986, doi: 10.1109/WCICA.2006.1713120.

[42]. P. -G. Ye, Y. -G. Wang, J. Li, L. Xiao and G. Zhu, "(T, ∈)-Greedy Reinforcement Learning for Anti-Jamming Wireless Communications," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9322486.

[43]. E. Eleftheriou and D. Falconer, "Adaptive Equalization Techniques for HF Channels," in IEEE Journal on Selected Areas in Communications, vol. 5, no. 2, pp. 238-247, February 1987, doi: 10.1109/JSAC.1987.1146531.