Cape Peninsula
University of Technology

Multiuser detection in hybrid Non-Orthogonal Multiple Access (NOMA) using machine learning

By

Mogomotsi Daphney Motsaathebe

Dissertation submitted in fulfilment of the requirements for the degree Master of Engineering in Electrical Engineering

In the Faculty of Engineering and Built Environment

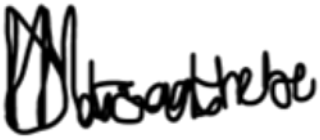
at the **Cape Peninsula University of Technology**

Supervisor: Prof V Balyan
Co-supervisor: Dr A Periola

Bellville Campus
Date submitted: 06 March 2025

DECLARATION

I, Mogomotsi Daphney Motsaathebe, certify that the contents of this dissertation are my own unassisted work and have not previously been submitted for academic examination towards any qualification. Furthermore, it expresses my own opinions rather than those of the Cape Peninsula University of Technology.



17 January 2024

Signed **Date**

ABSTRACT

The growing popularity of network applications has created a higher need for limited radio spectrum resources due to the increasing number of users varying for access to these finite resources. Efficiently managing the utilization of spectrum becomes increasingly crucial as demand rises. One effective strategy to tackle this challenge involves employing multiple access techniques. Among these approaches Non-Orthogonal Multiple Access (NOMA) emerges as a method, in this context. NOMA enables several users to utilize the frequency band through power domain multiplexing, which greatly boosts spectrum efficiency when compared to traditional orthogonal methods. However as more users access the limited radio spectrum, the task of multiple user detection (MUD) becomes more complicated leading to challenges in effectively distinguishing and decoding signals from various users. Dealing with these complexities requires techniques that can address non linearities and interference present, in these systems.

Artificial Neural Networks (ANNs) are one of the ways to handle these problems in an efficient manner. ANNs have the capacity of handling complex non-linear relationships can substantially enhance the accuracy for Multiple User (MU) in NOMA systems. This dissertation presents an ANN model explicitly for MUD in NOMA networks. Our method capitalizes on the power of deep neural networks to discern multi-user signals and filters them out which alleviated disadvantages in classical detection methods. Through training, the ANN recognizes the patterns and characteristics of received signals. This new method improves detection accuracy and spectrum utilization. The integration of ANNs with NOMA systems is a good innovation in wireless communication technology providing better performance for massive user scenarios.

The ANN-based multi-user detector performance has been carefully tested using simulated experiments. The results show that at epoch 2, the proposed technique has a mean square error (MSE) of 0.023103, thereby proving the efficiency of the neural network in solving the multi-user identification problem. This level of performance will give an idea of the capability of ANNs in solving the complexity introduced by NOMA systems and will act as a benchmark for future improvements. The efficiency of ANN in this context therefore shows that it can boost detection accuracy and optimize spectrum use, hence rendering a valuable edge over other previous methods. The research outcome obtained in support of the efficiency of ANN-based approaches and has established a concrete platform for further enhancement and use.

Keywords – Multi-user detection, Hybrid, NOMA, Machine Learning, wireless network.

ACKNOWDLEGEMENTS

Lots of efforts have been taken in this research. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

- I would like to thank my supervisor Prof Vipin for providing up to date information with regards to the research and for support in completing this research.
- I would like to express my gratitude towards my co-supervisor Dr Periola for constant guidance to complete this research and thanks for all your encouragement throughout the course
- I would like to express my special gratitude to my family for their kind co-operation and encouragement that helped me a lot.
- My appreciations also go to my friend and people who have willingly helped me out with their abilities.
- My sincere gratitude to Cape Peninsula University of Technology bursary to support me financially towards my master's degree.

DEDICATION

I dedicate this thesis to my mother Susan Motsaathebe and my late brother Noa Erasmus Motsaathebe, may your soul continue to rest in peace my brother.

TABLE OF CONTENTS

Contents

Table of figures

List of Tables

GLOSSARY

Abbreviations		Definition/Explanation
3D EPA		Three-Dimensional Expectation Propagation Algorithm
3GPP		3rd Generation Partnership Project
5G		5th Generation
6G		6th Generation
AmBC		Ambient Backscatter Communications
ANN		Artificial Neural Network
AUD		Active User Detection
AWGN		Additive-White Gaussian Noise
BD		Backscatter Device
BEP		Bit-Error Probability
BER		Bit-Error Rate
BN		Bayesian Networks
BP		Belief propagation
BPSK		Binary Phase Shift Keying
BS		Base Station
CBF		Conventional Beam Forming
CD		Code Domain
CDRT		Coordinated Direct and Relay Transmission
CE		Channel Estimation
CNN		Convolutional Neural Network
CPU		Central Processing Unit
CRE		Cell Range Extension
CRS-NOMA		Cooperative Relaying Scheme-NOMA
CSI		Channel-State-Information
CSS		Cooperative Spectrum Sharing
D2D		Device-to-Device
DAE		Denoising Auto-Encoder
DAG		Directed Acyclic Graph
DC		Difference of Convex functions
DCBF		Dolph-Chebyshev beam forming
DD		Device Data Discovery
DF		Decode and Forward
DIP		Deep Image Processing
DL		Deep-Learning
DNN		Deep-Neural-Network
DNN_MUD		Deep Neural Network Multi-user Detector
DSP		Digital Signal Processing
DT		Decision-Trees
DTL		Deep Transfer Learning
EC		Ergodic Capacity
ECSIC		Ergodic Capacity Successive Interference Cancellation
ELM		Extreme Learning Machine
EPA		Expectation Propagation Algorithm
EP-based		Expectation Propagation based
ESC		Ergodic Sum Capacity

FCM		Fuzzy C-Means
FDMA		Frequency Division Multiple Access
FFNN		Feed-Forward Neural Network
FTPA		Fractional Transmit Power Allocation
GA		Gaussian Approximation
GMAC		Gaussian Multiple Access Channels
GOLD		Generalized Orthogonal Learning codebook
GPU		Graphics Processing Unit
GSSK		Generalized Space Shift Keying
HetNet		Heterogeneous Network
HFA		Hard Fusion Algorithm
IDMA		Interleave-Division Multiple-Access
IoT		Internet of Things
ipSIC		Imperfect Successive Interference Cancellation
KPIs		Key Performance Indicators
LDS		Low-Density Spreading
LMMSE		Linear Minimum Mean Square Error
MAC		Multiple Access Channels
MEX		MATLAB Executable
MF		Matched Filter
MIMO		Multiple-Input Multiple-Output
ML		Machine Learning
ML		maximum likelihood
MMSE		Minimum Mean Square Error
mmWave		millimetre Wave
MN		Minimum Normalization
MPA		Message Passing Algorithm
MSE		Mean-Square Error
MUD		Multiple-User Detection
MUD_NN		Multi-User Detection Neural Network
MUSA		Multi-User Shared Access
NE		Naive Estimation
NN		Neural Network
NOMA		Non-Orthogonal Multiple Access
NS-3		Network Simulator version-3
OFDM		Orthogonal Frequency-Division Multiplexing
OMA		Orthogonal Multiple Access
OP		Outage Probability
PAPR		Peak-to-Average Power Ratio
PD		Power Domain
PDMA		Power Domain Multiple Access
pSIC		Perfect Successive-Interference-Cancellation
PSK		Phase Shift Keying
PSs		Phase Shifters
QAM		Quadrature Amplitude Modulation
QoS		Quality of Service
RAM		Random Access Memory
SC		Sparse Channel
SCM		Superposition Coded Modulation
SCMA		Sparse-Code Multiple Access

SD		Spherical Decoding
SD		Spherical Decoding
SIC		Successive-Interference-Cancellation
SIMO		Single-input, multiple-Output
SINR		Signal-to-Interference-plus-Noise Ratio
S-MAP		Sparsity-aware Maximum Aposteriori Probability
SNR		Signal-to-Noise Ratio
STBC-CNOMA		Space Time Block Coding – Cooperative NOMA
SVD		Singular Value Decomposition
SVM		Support Vector Machine
TDMA		Time Division Multiple Access
URLLC		ultra-reliable low-latency communications
UWs		Unique Word sets
ZF		Zero Forcing
ZTE		Zhejiang Telecommunication Equipment

CHAPTER ONE: INTRODUCTION

1.1.Introduction

This dissertation focuses on optimizing the spectrum utilization of multi-users in hybrid non-orthogonal multiple access (NOMA) based networks using machine learning. Machine learning can be used to optimize resource allocation and spectrum utilization in these wireless networks. The main benefits of using machine learning for resource allocation include improved performance and reduced complexity. The main challenges include the need for large amounts of data and high computing power' (Yaohua , et al., 2019). This project will develop a machine learning algorithm that uses data from multiple users to optimize network resource usage. The algorithm will be designed to maximize the resource usage efficiency of the network while ensuring that all users have access to the resources they need. The maximization of resource efficiency will be achieved by analysing each user's data and identifying patterns and trends that can be used to optimize resource allocation' (Yaohua Sun, et al., 2019). The goal of this project is to develop a machine learning algorithm capable of optimizing the resource use efficiency of a hybrid NOMA network, based on user characteristics such as priority, resource requirements, location, etc. and time of day. The goal is to improve the user experience and reduce costs for the service provider. The algorithm considers multiple users concurrently to optimize the network for all users, not just individual users.

1.2.Description of the research problem

The problems and the main challenges being faced are the complexity of the network and the need to balance the needs of multiple users. Another challenge is the dynamic nature of the network. Users' needs may change over time, so the algorithm will need to be able to adapt and adapt to these changes. Another challenge is the availability of data, as it can be limited or inconsistent. Finally, technical limitations or constraints may need to be considered when designing the algorithm. These challenges will need to be addressed to create an effective solution and are suitable options to be considered to optimize speed and bandwidth for multiple users, while minimizing interference and maximizing resource efficiency in the hybrid Non-Orthogonal Multiple Access network.

The importance of the problem

The increased demand for wireless communication has created a need for more efficient and effective wireless networks. Some of the reasons for this increased demand are the growth in use of mobile devices, increase in data usage, the rise of the Internet of Things, or the need for faster and more reliable connections. Existing solution, such as single-user detection can be slow and inefficient, and maximum likelihood detection can be computationally intensive, have limitations that make them unsuitable for certain scenarios. Therefore, it is important to develop new solution that can overcome these limitations and provide improved performance.

The proposed solution

The proposed solution is multi-user detection in hybrid Non-Orthogonal Multiple Access (NOMA) using machine learning. This solution is better than the existing solutions, by highlighting its advantages. For example, hybrid NOMA with machine learning based multi-user detection can be faster and more efficient than single-user detection, and it can be less computationally intensive than maximum likelihood detection. It can be more adaptable and flexible than the existing solution. This is because it can learn and adapt to new and changing conditions, rather than relying on pre-defined rules or parameters.

1.3.Aims and Objectives of this research

- To develop a model that can accurately predict user behaviour and network usage.
- To create an algorithm that can make real-time adjustments to the network based on user needs and resource availability.
- Design a method for minimizing interference and maximizing resource efficiency in a hybrid Non-Orthogonal Multiple Access network.
- Create a system that can be implemented in real-world scenarios with minimal cost and complexity.
- Optimize the hybrid NOMA network's spectral efficiency.
- Minimize hybrid NOMA networks' latency.
- Minimize the training time and computational complexity of the machine learning algorithm.
- Make the machine learning model easy to understand and interpret, so that it can be used by non-technical users.

1.4.Background

In recent years, the demand for wireless communication systems with high data rates has increased significantly. Non-orthogonal multiple access (NOMA) emerged as a potential solution to accommodate the growing number of users and data transmission. However, in a multi-user environment, NOMA suffers from interference between users, which can significantly affect communication performance. To solve this problem, multiple user detection (MUD) techniques have been introduced in hybrid NOMAs. In this context, machine learning (ML) algorithms have shown the potential to improve MUD performance in hybrid NOMA systems by predicting user signals and channel states. This project will discuss the concept of hybrid NOMA and the use of machine learning algorithms for MUD in hybrid NOMA' (Yaohua , et al., 2019). First, hybrid NOMA is a combination of ordinary orthogonal multiple access (OMA) and non-orthogonal multiple access (NOMA). It allows multiple users to share the same resource using OMA for some users and NOMA for other users. This is achieved by dividing users into groups and assigning orthogonal resources to some users in each group and non-orthogonal resources to the rest of the users. Hybrid NOMA has emerged as a promising way to improve system capacity compared to pure OMA or NOMA' (Wei, 2019).

Multi-user detection (MUD) is used to ensure the reception of signals in the case where multiple users transmit via a channel while minimizing interference in a multi-user communication system. The goal of the MUD is to estimate the signals transmitted from all users of the system. Two known MUD techniques are: Linear MUD and non-linear MUD. Linear MUD techniques, such as Linear Minimum Mean Square Error (LMMSE), Zero Forcing (ZF), and Minimum Normalization (MN), have low computational complexity but are limited in efficiency capacity' (Chuan, et al., 2017) . Non-linear MUD techniques, such as maximum likelihood (ML) and spherical decoding (SD), can achieve better performance but have high computational complexity' (Xiaojuan , et al., 2018). Machine learning (ML) algorithms have shown the potential to improve MUD performance in hybrid NOMAs. ML algorithms use data-driven models to optimize MUDs for NOMA systems. They can take advantage of patterns and trends in data that are ambiguous or only detectable by humans to predict unknown signals. By using these algorithms, the accuracy of detection can be improved without prior knowledge of the system ' (Yaohua , et al., 2019). One type of machine learning method is the artificial neural network (NN). NN models consist of several layers of nodes, whose trigger function applies the input signal to the transmission function. For

example, in a feed-forward neural network (FFNN), the input layer represents the input signals, and the output layer represents the final output. The hidden layers between the input layer and the output layer can optimize the weights between them by back propagation. NNs can generalize beyond their training data, making them well suited for MUDs in NOMA systems.

Support Vector Machine (SVM) is another ML method useful for realizing MUD in hybrid NOMA. SVM is a classifier that separates input data into two classes using defined decision boundaries. It is particularly useful for noisy data, and it is less prone to over fitting than other classifiers' (Yaohua , et al., 2019). Decision trees (DT) can perform classification and regression tasks. DT models work with data streams in the same way, using the defined properties of each observation to determine predictions. DT works based on a set of if/then rules. In the case of NOMA, the DT model determines the signal characteristics of each user based on the characteristics extracted from the user's signals. Bayesian Networks (BN) is another ML option for MUD in NOMA. BN models represent the probability distribution of a set of random variables. The learning process will provide information about the probability of having a certain signal or noise value, given the channel conditions. Naive Estimation (NE) can provide domain knowledge by allowing the user to specify previous probabilities. Clustering algorithms, such as k-means clustering, can be used for MUD in NOMA. Data is grouped or classified, or clusters based on similarities in the data. Grouped data will be classified as signal or noise based on grouping information. This type of algorithm is very useful for dealing with noisy data that is common in communication systems.

1.5.Research questions

The main research questions are outlined as follows:

1. In what way could a hybrid NOMA network's spectral efficiency be improved?
2. How can the latency of hybrid NOMA networks be reduced?
3. How can the training time and computational complexity of a machine learning algorithm be minimized?
4. What are the best methods for making a machine learning model easy to understand and interpret?

1.6.Research Hypotheses

The following five hypotheses will guide this research:

1. By incorporating user data and network resource availability into the machine learning model, it will be able to make real-time adjustments that improve network performance.
2. By optimizing the power allocation and scheduling in a hybrid NOMA network, can be interference minimised and resource efficiency maximised.
3. A system that uses open-source tools and a simple architecture can be implemented in real-world scenarios at a low cost and with minimal complexity.
4. By using a simple and efficient machine learning algorithm, the training time and computational complexity can be minimised.
5. By using simple and intuitive visualizations, the machine learning model can be made easy to understand and interpret.

1.7.Delineation of the research

Although this topic area is broad and it is interesting willing to learn more about machine learning, hybrid NOMA and multi-user detection, this project will only focus on

- Developing machine learning algorithm that can optimize the spectrum resource usage of a hybrid NOMA network.
- How Machine Learning could be used to analyse the data and make decisions about optimizing the coding and power allocation

1.8.Significance of the research

The research project is important because it will help to enhance spectrum resource utilization. It will also contribute to the ongoing research on machine learning algorithms for network optimization.

1.9.Expected outcomes, results and contributions of the research

The goal of this research is to optimize spectrum resource usage for users. The research will also aid in improving the overall quality of service for all consumers. This research is crucial since it will aid in ensuring a great user experience. Furthermore, it will aid in cost reduction and service provider efficiency.

The desired output of this research is a thesis or research paper presenting novel schemes to increase spectrum usage efficiency while improving network efficiency. These results will be achieved using a machine learning algorithm developed in this research. The algorithm must be scalable and able to manage large amounts of data. The algorithm must be robust and accurate, even in the presence of noisy or uncertain data. The algorithm must be computationally efficient, generalized to be applicable to other types of networks and applications. The results of this research should have important implications for the future of network optimization. Machine learning algorithms can be effective tools to improve network performance even in complex and dynamic environments. The algorithm developed in this research has potential applications in many fields, including telecommunications, Internet of Things, and smart cities.

One limitation of this research is that it requires a significant amount of data to train the algorithm. However, this limitation can be solved by using techniques such as transfer learning or data reinforcement. Project limitations include the need for accurate data and the potential for errors in the algorithm. The main risk is that the algorithm does not work as expected and cannot correctly identify the highest priority user. The main challenge is the need for accurate and up-to-date data about users and their connections.

1.10. Project Plan

To finish the planned research project, a two-year timeframe has been devised, commencing in February 2023 and ending in September 2024. The first year of the project will be devoted to the development of the research proposal, which will define the methods, aims, and expected outcomes of the research. The second year will be devoted to the design and production of the final dissertation, which will summarize the research findings and provide suggestions for future work in the subject.

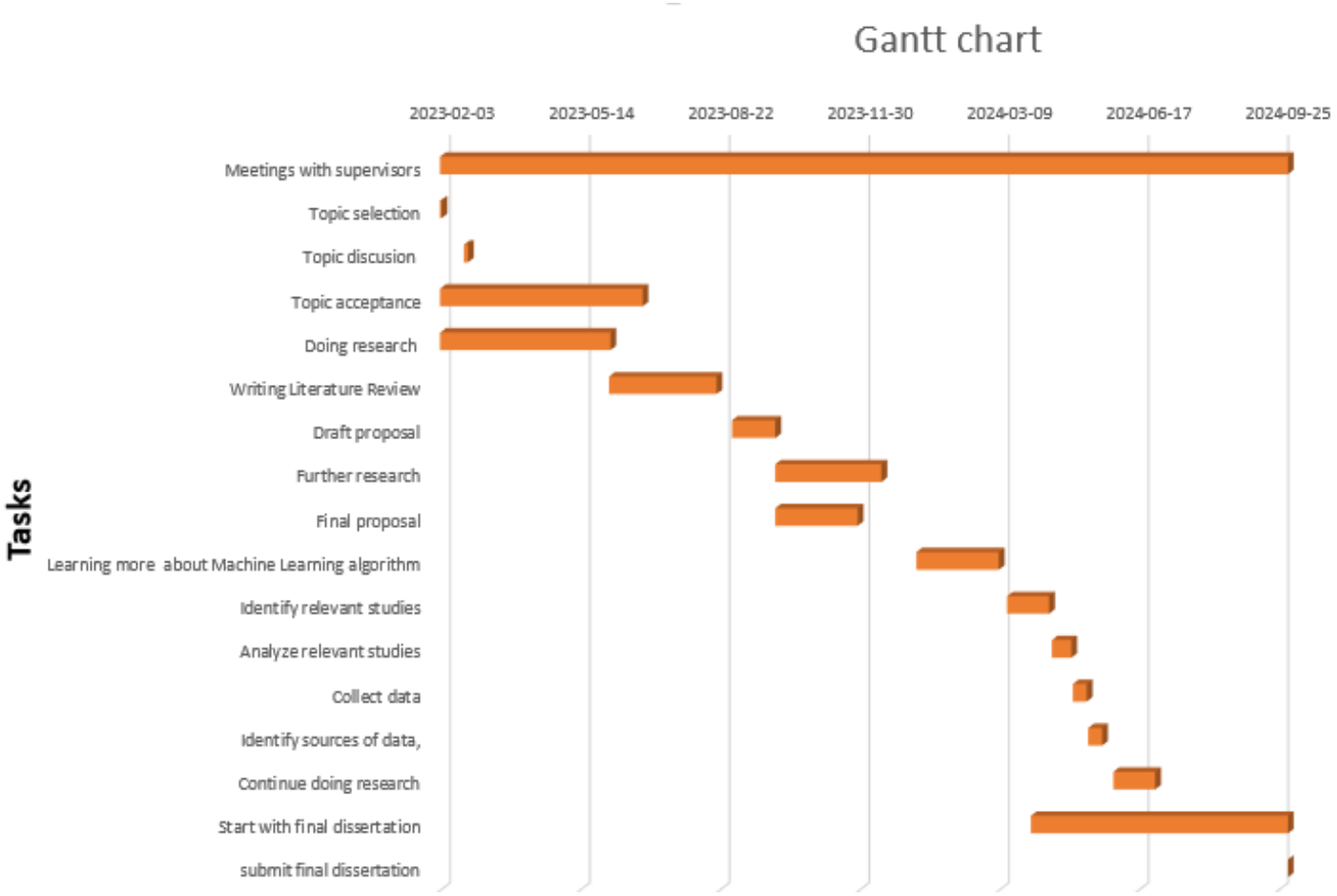


Figure 1.1: Project plan Gantt chart

1.11. Budget

Table 1.1 provides a detailed breakdown of the estimated expenses for the proposed research project, which will be fully funded by the student. This includes costs for materials, equipment, and software. All expenses are listed in detail, with a total cost estimate for the entire project.

Table 1.1: Estimated Expense

EXPENSES	**TOTAL**
Printing Material Paper and printer	R 5000
Laptop high specification	R30 000
Hard drive 1 Terabyte	R 5 000
Software package	R10 000
Total	R 50 000

CHAPTER TWO: LITERATURE REVIEW

This chapter reviews related literature on the problem of optimizing spectrum use in multi-user Hybrid NOMA networks in three ways: Non-Orthogonal Multiple Access (NOMA), Multi-User Detection (MUD), and Machine Learning (ML) applied to communication networks. The review assimilates recent research findings, highlighting notable advances, and incorporating gaps that the study at hand tries to bridge. Based on the analysis done regarding previous contributions, this chapter sets the ground for understanding how advanced techniques and methodologies can be used to improve the performance of a network and its spectrum efficiency in modern wireless communication systems.

2.1.Literature research papers used

The discussion in this aspect presents the spread of research literature that was reviewed in the proposed research. The spread of relevant research literature is presented in Figure 2.1. Figure 2.1 represents the relevant literature that was reviewed as part of this research's literature review. These publications were chosen for their relevance to the study topic and academic rigor.

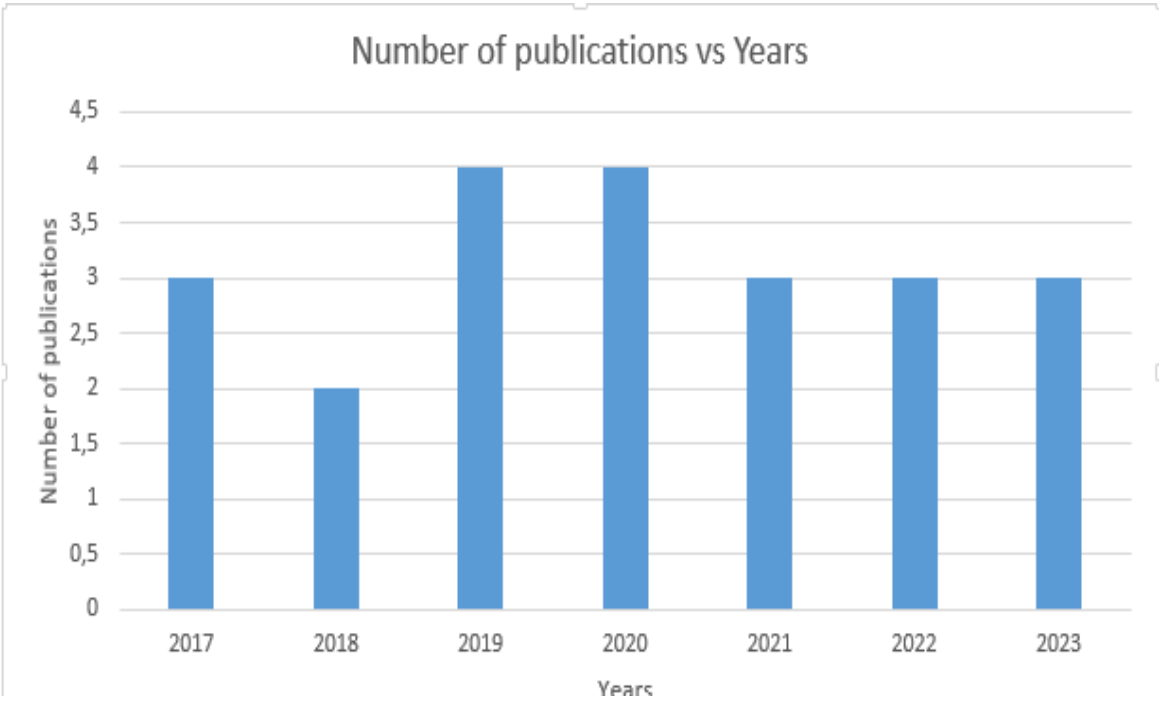


Figure 2.1: Graph shows the literature review papers used

2.2. Comparative analysis contribution made in projects related to Non-Orthogonal Multiple Access (NOMA)

The presented research examined the design of energy-efficient resource allocation for downlink NOMA networks' (Fang, 2017). Initially, the research examines resource allocation for an energy-efficient single cell NOMA network with perfect channel state information (CSI). It then extended its resource allocation strategy to the imperfect CSI case. The following are the outcomes: The challenge of energy-efficient resource allocation in a downlink NOMA wireless network by allocating just two users to the same sub-channel he problem under consideration is related to efficient resource management in Non-Orthogonal Multiple Access (NOMA) systems. To tackle this problem efficiently, it was divided into two main components: sub-channel allocation, and power distribution. This creates the optimization challenge, as some of the functions participate in the process are nonconvex. To solve this, the Difference of Convex (DC) optimization method was adopted. Therefore, to avert this, the problem was solved with only a sub-optimal distribution of power between the sub-channels. A Schema for allocating sub-channel power has been proposed. The numerical findings and evaluation demonstrated that averaging the suggested resource allocation scheme's average and energy efficiency performance OFDMA system is in place' (Fang, 2017).

The formalized probabilities are the mixed non-convex resource optimization problem was re-expressed as a non-stochastic problem for a NOMA system with imperfect CSI. An iterative allocation technique for power allocation was devised based on the recommended low complexity sub-optimal user planning. To enhance performance, NOMA HetNet uses energy-efficient single sub-channel allocation. A mapping approach with perfect CSI for downlink NOMA HetNet is proposed. A shared sub-channel and power allocation strategy in tiny cells can increase system energy efficiency. An optimum power allocation strategy in macro-cells is offered as a future upgrade. Another research project concentrated on a power-domain NOMA-based overlay spectrum sharing method' (Kader, 2019). This research project presents research that proposes a two-phase clustering-based Cooperative Successive-SIC (CSS- SIC) protocol that considers the NOMA forwarding strategy in both situations Phase over separate Rayleigh fading channels. The following is an analytical derivation of Energy Spectral Consumption (ESC) and Optimal Power (OP) using both Perfect Successive Interference Cancellation (pSIC) and Incomplete Successive Interference Cancellation (ipSIC). The suggested Cooperative Successive-SIC NOMA with Channel-Dependent Receive Thresholding (CSS- SIC NOMA-CDRT) performance increase is confirmed by simulation and analysis, and Monte Carlo simulation confirms the correctness of analysis results. The proposed CSS-NOMA-CDRT's

performance is tested for two cases: (i) Case-1 (C1) - successful decoding at relay, and (ii) Case-2 (C2) - failed decoding at relay' (Kader, 2019).

Wei (2019) developed two beam-width control approaches for wireless communications, namely conventional beam forming (CBF) and Dolph-Chebyshev beam forming (DCBF), which are applicable to scenarios with constant-modulus phase shifters (PSs) and amplitude-adjustable PSs, respectively' (Wei, 2019). More crucially, the primary lobe power losses caused by beam-width management were identified. They studied an asymptotically optimum analogue beam former design to minimize the main lobe power loss in a single-RF chain system, which provides as a basis for the design of analogue beam formers in multi-RF chain systems, based on the two beam-width control approaches. To create a stable user grouping approach, a NOMA user scheduling algorithm was developed for the proposed hybrid millimetre wave (mmWave) NOMA scheme employing coalition formation game theory. The energy-efficient digital pre-coder design is posed as a non-convex optimization problem using the derived NOMA user grouping approach to optimize system energy efficiency while considering the minimal individual user data rate need' (Wei, 2019).

The investigated issue is searching for the ways to increase the energy efficiency of the communication system by regulating the beam width. The investigation converts the problem into something easily tackled by iterative block coordinate ascent algorithms using quadratic transformations. This approach is then designed to reach a local optimum with respect to energy efficiency. Using these algorithms and conducting the simulation for a long time, it is proved that the better control on the beam width can better improve the energy efficiency of the system. This improvement is significantly more significant than the OMA and NOMA systems that do not use the beam width control' (Wei, 2019). The findings shows that the suggested broader analog beam-width technique is more resistant to beam alignment error than the basic approach' (Wei, 2019). Myo (2020) improved the system's achievable rate by using an OFDM-based CRS (Cooperative Relaying System)-NOMA. The impact of OFDM-based CRS-NOMA on various system performances and reduced receiver complexity are also investigated (Myo, 2020). Barbé (2020) explored how the NOMA-CAP 5G signal works and created a MATLAB application that can manage and simulate NOMA-CAP transmissions signal' (Barbé, 2020). This application enables the users to vary various parameters of the transmitted signals. The number of bands, signal bandwidth, modulation, and power ratio between users, for example, may be selected, and various parameters can be selected for different bands. To do this, the application

and sender/receiver code are used to generate/restore various NOMA-CAP signals for analysis in various simulation situations.

A Comparison Performance Evaluation of Hybrid NOMA Schemes for 5th Generation (5G) and Beyond Wireless Communications was focused on and investigated to determine the probability spread of the signal-to-interference-plus-noise-ratio (SINR) for the basic combinations that could be used to design and operate STBC-CNOMA devices (Bal, et al., 2021). The advanced knowledge that anticipated any 6-user signal under timing defects, perfect SIC, perfect CSI, and bit error probability in channel estimation for a promising STBC-CNOMA system are among the contributions of the STBC-based CNOMA program. An appropriate computational framework needed for the model is presented. To understand the results, evaluate the same contrast with other possible hybrid models for NOMA. This should be accompanied by Bit-Error Probability (BEP) calculations as well as energy and spectral efficiency trade-offs and SIC iterations as a function of users' (Bal, et al., 2021).

Some of the major development (REBHI, 2022) include the derivation of a universal denoising and decoding strategy for Sparse-Code Multiple Access (SCMA) detector architecture for enhancing the non-orthogonal multiple access strategies for massive communications. To begin, a denoising autoencoder (DAE) is trained to minimize the impact of his additive white Gaussian noise (AWGN) by attempting to eliminate it from the incoming signal. The denoising autoencoder receives a noisy sparse code multiple access (SCMA) signal as input. The output, on the other hand, is a noise-free approximation of the original output. The DAE's output is a noise-free approximation of the original signal. The DAE's purpose is to learn and recreate how to represent the sent data. Then, a deep neural network (DNN) is trained to recognize the transmitted bits in a denoised SCMA signal' (REBHI, 2022). End-to-end objective function training is performed on the two blocks of DAE-DNN detectors. The message passing algorithm (MPA)-based receiver must be aware of the codebook and channel status information.

As a result, earlier DL-based SCMA detectors and MPA are unfair. SCMA codebook knowledge is incorporated' (REBHI, 2022). The received vectors and the distance between each of the potential superposition combination codewords are used in the recognition procedure. First, this calculates the closest convolutional codeword vector; the chosen vector is the input to the DNN that will decode the sent user bits. The concept is that knowing the codebook leads to knowing the collection of vectors to which the noise-free received vector belongs, i.e. the overlay codeword's corrupted received vector is replaced by an approximated

noise-free one’ (REBHI, 2022). The purpose of this project made by’ (Omer Mohammed , et al., 2023): The goal of the Strategy for Non-Orthogonal Multiple Access and Performance in 5G and 6G Networks is to offer an overview of the important NOMA protocols for 5G and 6G networks that use code sharing as part of 3GPP standardization.

The goal of this article is to investigate and compare the many solutions presented to tackle the resource allocation problem to obtain high performance’ (Omer Mohammed , et al., 2023). Various firms have presented alternative NOMA ideas for 5G and 6G networks. NOMA is presently pursuing two major growth paths: The first is the Power division, while the second is the Code division. During the 3GPP standardization process, the developer's attention was concentrated on this second direction about the use of the NOMA scheme in 5G and 6G systems. Hardware communication, which is referred to as device-to-device (D2D) communication, is an integral part of the device communication process. This makes better use of network resources. Devices will be able to interact with one another directly, removing the requirement for a transmitting node. It is also one of the answers to the issue of restricted network coverage. This issue may be solved by utilizing D2D, which reduces charges and energy. One approach that is used to achieve this aim was to enlarge the network. They discussed NOMA technology and the significant benefits it has over traditional wireless technology. The most prevalent code-splitting NOMA variations, their features, and the possibilities and obstacles involved with implementing these variants were reviewed. The NOMA protocol allows wireless communication networks to grow significantly. In other words, 5G and 6G networks will boost network performance’ (Omer Mohammed , et al., 2023). Table 2.1 compares the methodology, achievements, and hardware/software employed in various research articles. This comparison serves as a foundation for the technique used in this study, as well as the selection of relevant methods and tools.

Table 2.1: Analysis of Non-Orthogonal Multiple Access (NOMA)

Paper (Text reference)	Aim of the paper	Methods used	Used hardware/Software	Achievements
Fang Fang, December 2017	Energy efficient resource allocation for Non-Orthogonal Multiple Access (NOMA) systems.	*Resource allocation method * Convex optimization method *Exhaustive searching method *Interior point method *Sequential quadratic Programming *Dual decomposition method *Gradient method	Difference of Convex (DC) functions Programming	According to simulation research, as compared to standard orthogonal multiple access (OMA) systems, the suggested resource allocation scheme for NOMA heterogeneous networks (HetNets) achieves optimized energy efficiency. Lastly, the study looked at the energy efficiency of NOMA HetNets with incorrect channel state information (CSI), and the simulation results were used to assess the performance. NOMA HetNets therefore achieves a higher energy efficiency because of being able to cancel out the interference resulting from imperfect CSI, which in OMA systems results in high levels of interference.
Md. Fazlul Kader, 4 December 2019	A Power-Domain NOMA based overlay spectrum sharing scheme.	*Fixed power allocation method		This work proposes and investigates an overlay spectrum sharing method based on clustering. In a coordinated direct and relay transmission (CDRT) situation, this method takes advantage of NOMA. Researchers evaluate the suggested cooperative spectrum sharing (CSS)-NOMA-CDRT in terms of outage probability (OP) and ergodic sum capacity (ESC)/ergodic capacity (EC). The likelihood that the signal-to-noise ratio (SNR) drops below a particular threshold and, as a result, the communication link's quality is insufficient is represented by the probability of failure/OP. Additionally, the Monte Carlo simulation is used to construct and validate the closed-form solutions for OP and ESC/EC. The construction of a NOMA variation to enhance the effectiveness of cooperative spectrum sharing in Cognitive Radio Networks (CRNs) has been the main topic of discussion. Here, the NOMA variation is used in a CDRT scenario to enhance system performance through the utilization of both direct and relay transmission. Comparing the suggested CSS-NOMA-CDRT to the standard methods, notable gains in throughput, OP, and ESC/EC are achieved.
Zhiqiang Wei, June 2019	Non-orthogonal Multiple Access Performance Analysis and Design for Wireless Communications	* Low-density spreading (LDS) method *Two-stage beam forming method *Worst-case method *Stochastic methods *Continuous Relaxation and Penalty Method *B&B method *Interior-point method *Channel estimation methods,	- Mosek - convex (CVX) programming	In a PhD thesis, Zhiqiang Wei investigated and addressed NOMA designs and performance studies in wireless communication systems. He demonstrated that NOMA could achieve significant performance gains over OMA in a variety of scenarios, including: (1) performance gain under different user pairing strategies, (2) performance gain under imperfect channel state information, (3) performance gain with cooperation, and (4) performance gain with beamforming.

Moe Ma Ma Myo, July 18, 2020	Employing OFDM-based NOMA system, a cooperative relaying system	*The hybrid relay selection approach and optimal power allocation *The Amplify and Forward (AF) technique *The method of Decode and Forward (DF) *The water-filling technique *The fractional transmit power allocation (FTPA) method	MATLAB	The simulation results in this work show that when a fair power allocation technique is applied, CRS-NOMA, a NOMA variant based on Orthogonal Frequency Division Multiplexing (OFDM) and cooperative relaying, performs better. The suggested Cooperative Relaying Scheme - Non-Orthogonal Multiple Access (CRS-NOMA) was used to download huge files in a Long-Term Evolution (LTE) network, and it outperformed standard techniques in terms of throughput and fairness. The proposed approach can be implemented utilizing the NOMA algorithm's successive interference cancellation (SIC) technique.
Marc Muro Barbé, May 2020	MATLAB application development to generate NOMA-CAP 5G signals	*NOMA-CAP *Orthogonal methods	MATLAB	Three different scenarios were studied to demonstrate the effectiveness of the NOMA-Compressive Amplitude Packing (NOMA-CAP) algorithm. Researchers assessed the impacts of power allocation ratio, modulation type, and symbol rate. It was shown that NOMA-CAP could be easily configured and used for generating signals in a real-world setting. The results of the study confirmed that NOMA-CAP is a viable solution for CAP-based communications
Anindya Bal, January 2021	A Performance Comparison of Hybrid NOMA Schemes for Fifth Generation (5G) and Beyond Wireless Communications	*GSSK methods *Diversity method and spatial multiplexing *SIC method *Multiple user detection (MUD) method *Traditional orthogonal channel estimation methods	Cellular Specified Software Network	To compare the performance of MIMO-NOMA, NOMA-GSSK, and STBC-CNOMA schemes, simulation results were presented in terms of Bit Error Probability (BEP), spectral and energy efficiency, and SIC incidence. In the context of integrating Power Domain (PD) NOMA with Code Domain (CD) NOMA, NOMA hybridization has not been examined. The performance of the hybrid Non-Orthogonal Multiple Access - Generalized Space Shift Keying (NOMA-GSSK) has not been compared with that of a hybrid emanating from PD NOMA and CD NOMA. In comparison to the other two schemes, the results demonstrate that NOMA-GSSK has the lowest BEP and the maximum spectral efficiency. In terms of energy efficiency and the amount of Successive Interference Cancellation (SIC) occurrences, Space Time Block Coding - Cooperative Non-Orthogonal Multiple Access (STBC-CNOMA) clearly surpasses the other two hybrid NOMA systems.

Manel REBHI,26 April 2022	Contributions to NOMA techniques for massive communications	*Deep Learning methods *Belief propagation (BP) or SIC method *ML methods *Sparse-channel-based EPA (SC-EPA) method *SCMA detection methods *Existing channel estimation methods *Data-aided estimation method	40nm CMOS technology.	Assigned irregular codebooks for each group of users to achieve higher overall throughput when if users have different channel states. Simulations confirmed that using an adaptive SCMA design can achieve better performances compared to a conventional regular SCMA design.
Omer Mohammed Khodayer Al-Dulaimi, Aymen Mohammed Khodayer Al-Dulaimi, Maiduc Osiceanu Alexandra and Mohammed Khodayer Hassan Al-Dulaimi, 3 February 2023	Non-Orthogonal Multiple Access Strategy and Performance in 5G and 6G Networks.	*Revaluation of traditional methods *Multi-user reception method *SCMA method *Zhejiang Telecommunication Equipment (ZTE's) Multi-User Shared Access (MUSA) method *Power Domain Multiple Access (PDMA) method	*D2D communication *Software-configurable ad hoc interface for 5G systems,	To solve these concerns, it was recommended that NOMA technology be adopted. A substantial amount of effort has recently been put into researching ways to make 5G networks more efficient by utilizing NOMA technology. Many various forms of NOMA have been suggested as part of the 3rd Generation Partnership Project (3GPP) standardization process, but no definitive decision on the regulation of any NOMA scheme has been made.

2.3.Comparative analysis contribution made in projects related to Multi-User Detection (MUD)

The previous discussion has explained different aspects of multiple access techniques with focus on stating that NOMA with its various variants outperforms OMA and its variants, the focus has been on improving channel capacity, power efficiency and other quality of service (QoS) related measures in 5G and 6G networks. The next section focuses on another aspect of NOMA, this aspect is multiuser detection, and it will discuss and address the multiuser detection (MUD) problem.

The main contributions that' (GUIYONG, 2017) implemented in his project of compressed sensing based block type multi-user detection (CS-BT-MUD) for sporadic IoT, proposed CS-BT-MUD algorithm for block type compressed sensing multi-user detection (CS-MUD) problem, took an advantage of the block-type activity detection problem. A pre-processing

method was also proposed for complex-valued data that allows the process complex-valued data use the traditional Compressed Sensing (CS) algorithm. For numerical evaluation, he provided two practical traffic models. Basically, his various IoT applications are of great research interest. In recent years, these technologies will change lifestyles' (GUIYONG, 2017). For more critical MAI, which is more important in physics layer, is left for future work.

In (Zhao, et al., 2018) a compressive sensing-based multi-user identification technique for single-input multiple-output non-orthogonal multiple access (SIMO-NOMA) systems with multiple antennas is suggested' (Zhao, et al., 2018). The compressive sensing hard fusion algorithm (CS-HFA) suggested algorithm uses compressed sensing to accomplish multi-user detection (MUD). It gathers user activity by getting information using the usual CS algorithm at each antenna, after which it integrates and derives the observed user activity information of the active user set. Finally, the active user set that was collected may be utilized to estimate active user data. The proposed CS-HFA algorithm, which is intended for a grant-free, low complexity SIMO-NOMA system, outperforms standard CS-based MUD algorithms for NOMA systems with only one antenna at the base station (BS)' (Zhao, et al., 2018).

Sparsity-Aware Multi-User Detection for Massive Machine-Type Communications is the focus of this thesis' (Ahn, 2019), which proposes two novel sparse-aware multi-user detection approaches. First, a novel sparsity-aware ordering SIC technique for massive machine-type communications (mMTC) systems is presented. The proposed compressive sensing hard fusion algorithm (CS-HFA) for a grant-free, low complexity single-input-multiple-output non-orthogonal multiple access (SIMO-NOMA) system finds the best detection order using successive approximation (SA) based systems. When compared to the sequential interference cancellation (SA-SIC) approach, numerical simulations reveal that the CS-HFA algorithm greatly improves performance. (Ahn, 2019). After that, a Bayesian shared active user detection (AUD) and channel estimation (CE) approach based on expectation propagation (EP) is suggested for mMTC systems. It is a non-Bayesian strategy that does not make optimal use of previous statistical distributions of user activity and channels to increase work motivation based on observations that underpin typical CS-MUD systems.

The suggested approach iteratively selects the best fitting Gaussian distribution that is close to the posterior distribution of the composite goal vector of user activities and a channel by leveraging the prior distribution. The maximum a posteriori (MAP), (Xiaojie Wang, et al., 2020) an approximation approach is used to address the estimate issue, and active user detection

(AUD) and channel estimation (CE) are performed concurrently'. It is then acted upon depending on the insights gathered about user activity and channels from Active Device Data Discovery (DD). Numerical simulations demonstrated that the new technique is effective, greatly improving AUD, CE, and DD performance over the previous sparse recovery algorithm. Achievable Rate Region via Low-Cost Gaussian Approximation for Iterative Multi-User Detection proposed' (Xiaojie Wang, et al., 2020), a low-cost Gaussian Approximation (GA)-based MUD capable of near-optimal performance. It has been shown to boost performance for Gaussian signalling. The loss caused by discrete modulation as opposed to Gaussian signalling may be arbitrarily decreased using overlapping superposition coded modulation (SCM) technology. For forward error correction (FEC) algorithms tailored for a single user channel, this may not be the case.

MUD is an excellent alternative for MAC with a low-cost GA-based solution. FEC codes must be strictly followed; it is intended to be MUD compliant and to facilitate iterative detection (Xiaojie Wang, et al., 2020). A multi-user area selected for the EXIT diagram for code design implies that it is the overall rate capacity. The potential function of the MSE vector field created by "V" results in path independence' (Xiaojie Wang, et al., 2020). The maximum multiple access channel (MAC) capacity range is one FEC code per user at all locations. This avoids the SIC's related losses framing. The results presented above may be applied to multiple-input multiple-output (MIMO) MAC channels. The simulation findings demonstrate that a well-designed interleave division multiple access (IDMA) may reach this total rate MAC capacity of various decoding pathways within 1 decibel (dB) in the mean square error (MSE) vector field (Xiaojie Wang, et al., 2020).

The studies in' (JEON, 2021) compares four techniques for determining the set of active sequences. A correlation-based energy detector is used to estimate the most likely sequences. Similarly, the user can combine the following bits to shorten the length of B and get 2 bits. Combine energy detectors to determine the active sequence. The third method used is to create a histogram from the matched filter (MF) for stacking data into neural network input. In this neural network models, the structure is a fully connected layer with two inputs. Another input is Energy detector value' (JEON, 2021). A final method is to focus on distribution from MF to any user can compare the probability functions of active and inactive sequences. All systems are used to detect active sequences, and the above-mentioned minimal mean square error (MMSE) estimate is used to recover the original signal. Reversing the size of the whole MMSE

matrix, on the other hand, is difficult. For this, we used the clustering approach to minimize matrix size and achieve low time complexity' (JEON, 2021).

This thesis's primary contributions' (Peisen Wang, et al., 2022) First, a unique factor-graph model for code domain NOMA was developed to define symbol spreading and asynchronous interference at the same time. Time delay is added as an additional dimension to the typical bipartite factor-graph in this approach. The resulting 3D factor-graph is then partitioned into many connected sub-factor graphs, each of which is aligned to a different user's NOMA symbol. Meanwhile, two message passing techniques are presented to define the probabilistic interactions inside and across sub-factor graphs, namely intra-slice message passing and inter-slice message passing. Second, in the 3D factor-graph, an expectation propagation mechanism was constructed, and a unique low complexity multi-user detection approach, called the three-dimensional expectation propagation algorithm (3D-EPA), was proposed' (Peisen Wang, et al., 2022).

A two-step technique is presented, consisting of per-user estimate and inter-user message transfer' (Peisen Wang, et al., 2022). The complex symbol estimations are translated into a Gaussian distribution during the per-user estimation stage. During the inter-user message forwarding stage, symbol estimation is performed using information from the nearby sub-factor graph. Because it may take advantage of geographical and temporal connections between users, this method outperforms classic Bayesian minimal mean square error techniques. In addition, the researchers expanded 3D EPA to numerous antenna systems' (Peisen Wang, et al., 2022). Finally, to validate the bit error rate (BER) performance of 3D EPA with asynchronous NOMA, create a state evolution analysis and run Monte Carlo simulations' (Peisen Wang, et al., 2022). The suggested asynchronous code domain NOMA with 3D-EPA outperforms the synchronous one, especially under severe congestion, and outperforms simple common asynchronous detection techniques. The concepts of complexity and convergence are also examined.

To examine a large-scale, low-power Internet of Things (IoT) scenario, a research team developed a symbiotic communication model for an uplink NOMA system with direct connection between numerous base devices (BDs) and a carrier transmitter (CT)' (Chaowei Wang, et al., 2023). In this setup, the CT communicates its own data while simultaneously powering the BD via a direct link. Each BD can be in one of two states: energy harvesting or backscattering. The BD does not transfer any information when gathering energy. The researchers leveraged the symbiotic system's collaboration to jointly develop the transmission

signal waveforms of the CT and the BD. The developed UW selection technique was used to construct unique word sets (UWs) based on the strong correlation of generalized orthogonal learning (GOLD) codebook sequences' (Chaowei Wang, et al., 2023). Each non-orthogonal user had a unique UW. It presented a multi-user identification technique based on successive interference cancellation (SIC). The receiver in the symbiotic AmBC system expects that it knows the UW set of the CTs and BDs, as well as the overall number of BDs. However, because BDs might be power harvesting, the receiver must identify the number of active BDs. The proposed UW selection algorithm could guarantee the correlation performance of UW sequences, and the multi-user detection algorithm could improve detection performance on different links, different numbers of BDs, and different power reflection coefficients of BDs, according to simulation and analysis. It was also established that the approach was applicable to a variety of UW sequence lengths' (Chaowei Wang, et al., 2023). Table 2.2 contrasts the method, achievements, and hardware/software used in different study studies. This comparison is the basis for the strategy employed in this work, as well as the selection of applicable methodologies and instruments.

Table 2.2: Analysis of Multi-User Detection (MUD)

Paper	Aim of the paper	Methods used	Achievements
ZHANG GUIYONG**,** February 2017	Multi-User Detection Using Compressed Sensing Based Block Type for Sporadic IoT Communications.	*Conventional real-valued method *Complex-valued data pre-processing method	The main challenges for Internet of Things (IoT) communication were studied. Specifically, an algorithm was introduced, termed as the compressive sensing block type multi-user Detection (CS-BT-MUD) algorithm, to solve the MAI in large-scale IoT applications. While previous research focus on massive access have been focused on single type CS-MUD problem and have given a new CS based algorithm for block type CS-MUD problem which exploits the block sparsity nature in the activity detection. This is because of the loose coupling between the CS-BT-MUD and different hybrid NOMA variants that may arise.
Xiaojuan Zhao, Shouyi Yang, Aihua Zhang\S, Xiaoyu Li, 2018'	A Multi-user Detection Algorithm Based on Compressive Sensing for SIMO-NOMA Systems.	*CS-HFA detector	The suggested Compressive Sensing-based Hard Fusion Algorithm (CS-HFA) is especially proposed for the Single-Input Multiple-Output Non-orthogonal Multiple Access (SIMO-NOMA) communication network, according on simulation findings. This approach extends the CS-BT-MUD technique by loosely coupling the considered NOMA variant. SIMO-NOMA is intended for use in a terrestrial wireless network. Does the MUD proposed work in the MIMO-NOMA case.
Jinyoup Ahn FEBRUARY 2019	Multi-User Detection with Sparsity for Massive Machine-Type Communications	*EP-based joint AUD and CE method *Compressed sensing based multi-user detection (CS-MUD) *Sparsity-aware Successive interference cancellation (SA-SIC)	Numerical simulations show that the suggested technique enhances the performance of Sparsity aware successive interference cancellation (SA-SIC) significantly. SA-SIC was developed to minimize the complexity associated with previously recognized successive interference cancellation technologies such as compressive sensing - multiuser detection and sparsity-aware maximum aposteriori probability (S-MAP) based multiuser detection. Then, for mMTC systems, an EP-based Bayesian joint active user detection (AUD) and channel estimation (CE) approach is provided.
Xiaojie Wang, Chulong Liang, Li Ping, and Stephan ten Brink, MAY 2020	Rate Region Achievable for Iterative Multi-User Detection Using Low-Cost Gaussian Approximation	*MSE evolution method *LDPC code design methods	The simple interleave-division multiple-access (IDMA) has been shown to be optimal under Gaussian approximation (GA), relying on a low-cost GA-based multi-user detector (MUD), is capacity-achieving for general Gaussian multiple access channels (GMAC) with an arbitrary number of users, power distribution, and single or multiple antennas. In the mean-square error (MSE) vector field, IDMA with matching codes achieves capacity for every given decoding path. The presented research aims to show that the use of the gaussian approximation IDMA technique does not lead to a significant degradation of the QoS for networks using the IDMA technique.
NICHOLAS JEON, May 2021	Active user detection and low complexity multiuser detection for Unsourced multiple access	- clustering method - correlation-based energy detector	Two methods were presented for this process, with a different approach to reduce the complexity. The spreading sequences used in this research are Gaussian random variables. Furthermore, the error performance of this technique is significantly reliant on the spreading sequence and channel code lengths chosen. The purpose of this work is to create a low-complexity MUD competent system. Only one active user is found in this situation. Energy Detector (Sensing), Neural Network, and Hypothesis testing methods are approaches used for the detection of the active user.

Peisen Wang, Neng Ye, Jianguo Li, Boya Di, and Aihua Wang, October 2022	Expectation Propagation across a 3D Factor-Graph for Asynchronous Multi-User Detection in Code-Domain NOMA	*Matching method *Frame-level iterative Method *Low-complexity and efficient inference method	Lastly, according to the link level simulations, the following observations were made: For starters, the bit error rate (BER) performance of asynchronous code-domain NOMA with the 3D expectation propagation algorithm (3D-EPA) can beat that of its synchronous counterpart, particularly under high overload conditions. Second, the linear complexity of the 3D-EPA is comparable to that of classical EPA. Furthermore, the acquired BER performances with varied frame lengths and spreading factors, as well as the convergence performance of the 3D-EPA, were shown.
Chaowei Wang, Mingliang Pang, Gaofeng Cui, Xinshi Chang, Fan Jiang, Yuan Yao, Weidong Wang, 2023	Multi-user Detection and Joint Waveform Design in Symbiotic Ambient Backscatter NOMA Systems.	*Data detection method *Stochastic optimization method *Single-threshold detection method *Modulation and coding methods	The conventional correlation detection algorithm was improved to overcome the single threshold constraint by adding a second one, while using SIC to reduce multi-access interference. Simulation results show that the proposed Ergodic Capacity Successive Interference Cancellation ECSIC outperforms the conventional algorithm at low SNR for two backscatter devices (BDs) scenario. In addition, it is validated that the proposed evolved correlation detection algorithm based on ECSIC can still achieve improved detection performance even without direct link, and the system capacity of Ambient backscatter communications NOMA (AmBC-NOMA) does overcome AmBC-OMA, even though the thresholds in the ECSIC must be updated periodically.

2.4. Comparative analysis of contributions regarding the use of Machine Learning in Communication Networks

Machine learning and communication technologies are convergent disciplines. When paired with modern machine learning approaches, today's communication systems create massive volumes of data. The use of the resulting data enhances the design and administration of communication networks and components' (Wojciech Samek, et al., 2017). Furthermore, freshly designed end-to-end training techniques open new avenues for collaboratively optimizing communication system components. Furthermore, machine learning methodologies are critical in many developing communication technology application areas, such as smart cities or the Internet of Things. The discussion by' (Wojciech Samek, et al., 2017) presents an overview of the use of machine learning in many sectors of communication and explores two sample wireless network applications. It also selects interesting future research areas and examines their potential significance.

Jingjing Cui, et al., (2018) explore the problem of improving the overall speed of the millimeter-wavelength NOMA system by combining the user clustering approach, the beamforming and the energy allocation strategies. The optimisation problem is complex in this case because it is a combinational problem, therefore trying to find an overall solution is hard (Jingjing Cui, et al., 2018). To minimize complexity, the optimization challenge is subdivided into multiple subproblems, and the appropriate solutions of those subproblems are then

expanded. The user millimetre wave (mmWave) channel feedback correlation function was used to build NOMA diagrams using the new K-means-enabled machine learning framework. Furthermore, inspired by the dynamic characteristics of users' needs, an online user clustering algorithm using the K-means algorithm to match newly arrived users in the system was proposed. This can significantly reduce the computational complexity in comparison to a strategy in which the clustering structure is completely changed even if only one new user enters the system' (Jingjing Cui, et al., 2018).

Under the assumption that the BS distributes its power evenly among multiple clusters, a closed-mode optimum power allocation technique was developed' (Jingjing Cui, et al., 2018). It should be noted that the suggested source allocation technique is appropriate for optimizing the total throughput of other NOMA situations with minimum throughput limits, such as multi-cell NOMA scenarios. Collaboration to provide an efficient solution framework for NOMA. The simulation shows that the proposed K-mean machine learning framework for mmWave NOMA systems outperforms existing techniques based on matching theory. Furthermore, our proposed approach can match the performance of an ideal set of users' (Jingjing Cui, et al., 2018). The basic concepts and wide applications of common machine learning techniques used in wireless networks are outlined in detail, including supervised learning, unsupervised learning, reinforcement learning, neural network (deep), and transfer learning, because of its growing relevance in wireless communication systems, NN (deep) and transfer learning are indicated separately' (Yaohua Sun, et al., 2019).

The study by (Yaohua , et al., 2019) covers all levels except the well-researched physical layer and provides a complete analysis of the literature on applying machine learning to resource management, networking, mobility management, and localization. Power control, spectrum management, beam generator design, link management, buffer management, and compute resource management are the resource management applications, and the network is split into user association, BS switching control, routing, and clustering. Furthermore, the publications assessed in each application area are categorized based on machine learning techniques used and cover the bulk of network scenarios that may develop in the 5G future, such as media networks, small mobile networks, and cloud radio access networks' (Yaohua , et al., 2019). Several requirements for using machine learning in wireless networks are identified, including issue categories, learning data availability, time cost, and so on.

Furthermore, conventional procedures cited in research articles are discussed, as are performance comparisons with machine learning-based solutions based on dynamic clarification' (Yaohua Sun, et al., 2019). Future problems and unresolved concerns associated with the implementation of machine learning in wireless networks are highlighted in relation to machine learning-based network slicing, the standard dataset for future study. The thesis presents an in-depth, methodical, and thorough examination of the role of machines and deep learning techniques in IoT.' (Fatima , et al., 2020). The researchers report the advanced ML and DL results in IoT networks, with an emphasis on IoT network security and privacy. The researchers outline the shortcomings of present IoT network security solutions, which need the application of ML and DL approaches. The researchers also give an in-depth analysis of several research problems that must be solved in the deployment of ML and DL techniques in IoT networks' (Yaohua Sun, et al., 2019).

Ohtsuki (2022) highlighted several machine learning approaches and applications in 6G wireless communication, with an emphasis on the physical layer' (OHTSUKI, 2022), and it demonstrated end-to-end communication system learning using neural network-based autoencoders. Then, for massive multiple-input (MIMO) outputs, introduce various machine learning algorithms. To identify huge MIMO signals, a neural network-based trust belief propagation (BP) technique was developed. This approach is built on the notion of deep implementation, which involves implementing iterations of the inference algorithm in a layered neural network structure. The authors also provide a signal identification method based on the BP algorithm, as well as a denoising technique based on deep learning (DL) and earlier deep image processing (DIP)' (OHTSUKI, 2022), because the number of antennas is significant in big MIMO systems, as the name implies, the estimated number of channels is also important. The length of the pilot signal is restricted due to the channel's timing characteristic; hence the length of the orthogonal pilot signal is finite. As a result, the same pilot signals are repeated in neighbouring cells, lowering channel estimate efficiency; this flaw is known as pilot pollution. Two neural network-based strategies were proposed by the authors to lessen the impact of pilot contamination. The research also looked at channel state feedback (CSI), which may be an issue in big multiple-input multiple-output (MIMO) systems due to the volume of feedback. The authors described a neural network-based CSI feedback method as well as deep transfer learning (DTL)' (OHTSUKI, 2022). Table 2.3 compares the methods, and achievements employed in various research projects. This comparison serves as the foundation for the strategy used in this study, as well as the selection of relevant techniques and instruments.

Table 2.3: Analysis of Machine Learning (ML)

Paper	Aim of the paper	Methods used	Achievements
Wojciech Samek, Slawomir Stanczak, Thomas Wiegand**,**13 Oct. 2017	The combination of machine learning and communication	Machine learning methods.	The rising mutual effect of machine learning and communication technologies was examined in this study. Learning algorithms have been shown to excel not only in traditional network management tasks such as routing, channel estimation, and peak-to-average power ratio (PAPR) reduction, but also to be a key component of many emerging communication technology application fields such as smart cities and the Internet of things.
Jingjing Cui, Zhiguo Ding, Pingzhi Fan, and Naofal Al-Dhahir, NOVEMBER 2018	User Clustering Using Unsupervised Machine Learning in Millimetre-Wave-NOMA Systems	* Zero-forcing beam forming method *Elbow method	Channel correlations are a useful statistic for performing K-means clustering. The researchers also created a closed form formula for the best power allocation for each cluster, assuming equal power distribution across clusters. The suggested K-means enabled machine-learning framework for mm-wave-NOMA systems outperformed mm-wave-OMA systems in simulation.
Yaohua Sun, Mugen Peng, Yangcheng Zhou, Yuzhe Huang, and Shiwen Mao, FOURTH QUARTER 2019	Machine Learning in Wireless Networks: Key Techniques and Unresolved Issues	*Diverse machine learning Methods * Gradient descent (GD) methods * Cell range extension (CRE) method * State of the art method * Q-function Learning method	This study covers the state of the art in machine learning (ML) applications for wireless communications and identifies numerous outstanding issues. Given the complexities of these applications, the researchers split the area of expertise into three categories: resource management at the medium access control (MAC) layer, networking and mobility management at the network layer, and application layer localization. Several ML-based techniques have been developed within each of these categories to enable wireless networks to run intelligently.
Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain, THIRD QUARTER 2020	Machine Learning in IoT Security: Current Issues and Future Prospects	* ML and DL methods *Unsupervised learning methods *Extreme learning machine method (ELM) algorithm *Leverages fuzzy C-means (FCM) * Convolutional Neural Network (CNN) based method	This article examined the role of machine learning (ML) and deep learning (DL) on the Internet of Things (IoT) from the standpoints of security and privacy. The issues of security and privacy on the Internet of Things, as well as attack vectors and security needs, have been examined. Several ML and DL approaches, as well as their applicability to IoT security, have been described. Traditional ML mechanisms' weaknesses have also been highlighted. Finally, existing security solutions, open issues, and future research possibilities are discussed.
Tomoaki OHTSUKI, Fellow **,**2022/08/10	In 6G Wireless Communications, Machine Learning	*Cognitive spectrum sharing methods * Linear detection Methods *Gradient-based methods. *Machine Learning method *Deep transfer learning method	The purpose of this study was to provide an overview of certain machine learning (ML) techniques and applications in 6G wireless communications, with an emphasis on the physical layer. The dynamic contexts in which wireless communications operate are one of the problems in applying ML to real-world systems. Based on data, ML develops inferences and predictions, and wireless environments change dynamically. As a result, using ML approaches in such situations might be difficult.

2.4.1. Evaluation computing costs and software costs for cost optimization

To maximize the efficiency of the budget that is allocated for research in machine learning and advanced wireless communications, effective management of computational resources and software expenditures is a necessary requirement. The purpose of this first evaluation is to draw comparison of the estimated expenses related to different computer hardware and software that were used in the studies recently, thus providing some insights to the potential areas for cost savings. These costs are how researchers and institutions can, firstly, make informed resource allocation decisions, and, secondly, detect the possible ways to save the money without the quality of research being affected.

Table 2.4: Evaluation of computing hardware and software costs

Paper	Aim of the paper	Computing Hardware costs	Software costs
Wojciech Samek, Slawomir Stanczak, Thomas Wiegand,13 Oct. 2017	The combination of machine learning and communication	R89,204.94 for GPU cluster	R21409.18 for MATLAB license **NS3**: free
Jingjing Cui, Zhiguo Ding, Pingzhi Fan, and Naofal Al-Dhahir, NOVEMBER 2018	User Clustering Using Unsupervised Machine Learning in Millimetre-Wave-NOMA Systems	R53,522.96 for server	R21409.18 for MATLAB license NS3: free
Yaohua Sun, Mugen Peng, Yangcheng Zhou, Yuzhe Huang, and Shiwen Mao, FOURTH QUARTER 2019	Machine Learning in Wireless Networks: Key Techniques and Unresolved Issues	R124,886.91 for high-performance computing nodes	OMNeT++: free R21409.18 for MATLAB license
Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain, THIRD QUARTER 2020	Machine Learning in IoT Security: Current Issues and Future Prospects	R71,363.95 for network simulation hardware	OMNeT++: free R21409.18 for MATLAB license
Tomoaki OHTSUKI, Fellow	In 6G Wireless Communications, Machine Learning	R107,045.92 for experimental setup	OMNeT++: free R21409.18 for MATLAB license **NS3**:

An explanation of the costs of the equipment's in Table 2.4 demonstrates that super-computing nodes are the mostly costly while Graphics Processing Unit (GPU) clusters are in the middle range. It is therefore vital for entities to carefully assess the validity of complex computational architectures expenses and seek for equivalent cost-effective means of fulfilling the same operational outputs.

In expense reduction one of the strategies would be to review the necessity of harnessing expensive hardware tools for the research work. Through the examination of less expensive hardware possibilities and combining it with an optimum setup and choice of hardware parts, substantial cost reductions can be made. Further, the use of openness and free edition software like OMNeT++ and NS3 can be used with optional than to incorporate with costly licenses. These tools offer effective functions of network simulating and analysing beyond the need of using expensive commercial software.

This process involves the management of resources as well, it is also one of the vital components of this process. Performing these analyses lets the researcher determine how his or her resource limitations match overall goals, as well as how research budgets can be maximized. However, by properly coordinating the costs with the goals of research, the researchers can be able to properly control the resource expenses and the budget.

In this dissertation, it is imperative to note that the research is centred on developing a multi-user detection that may not afford expensive computing equipment. This consideration gives focus on the issue of cost and points to the fact that appropriate and affordable equipment along with proper software should be chosen depending on the cost capacity of the research. This aspect is addressed before designing the research to ensure that the design is operationally feasible within the available resources.

CHAPTER THREE: METHODOLOGY

This chapter discusses research methodology in the design of Hybrid Non-Orthogonal multiple Access (NOMA) networks to support many users while using limited available spectrum utilization and incorporating machine learning. Research is intended to enhance network performance and efficiency through producing synthetic data and machine learning methods as well as artificial neural network (ANN) multi-user detector. These stages include data collection and machine learning models' application, system architecture and assessment. This is centred on enhancing user experiential and spectrum efficiency in wireless network systems through activities that incorporate real- life scenarios and artificial intelligence.

3.1. Research design and methodology

The research methodology is structured into the following parts:

1. Data Collection and Preparation
2. Machine Learning Model Development
3. System Architecture Design
4. Performance Assessment

3.2. Data Collection and Preparation

The goal is to use synthetic data for two main purposes: uses in education and in experiments and training of different machine learning algorithms and models.

Synthetic Data Usage:

Educational Purposes: Information obtained enables the use of synthetic data, into reproducing real-life conditions within educational contexts. They enable students and researchers to grasp different subjects and networks that in fact don't need data, which might not be easy to come by, or process. For instance, synthetic data is applied in teaching examples that may include a class assignment or a workshop to explicate machine learning algorithms or data analysis approaches.

Training Machine Learning Models: It is a simulation that helps the practitioners work on building machine learning models on data that has been artificially created. It enables the

researchers and practitioners to experiment with the algorithms in conditions and scenarios which can be so complex as to be impractical with real data. This can be one or several corner cases; some scenarios that happen rarely; some configurations which are important for the model's performance and stability assessment.

Testing Machine Learning Models: Hence synthetic datasets are also useful tools in assessing and identifying the effectiveness of pre-specified machine learning algorithms. They can help evaluate how effectively a model performs on possibly diverse types of data or circumstances under which it can be applied that were not realised during the training process. Through Synthetic data, Researchers gets to know about the issues that may arise when the model is implemented for Real-world applications so that they can rectify them and make a more refined model to be implemented in Real world.

1. Data Collection:

- User Data: Acquire user details such as User-id, location or connection requirements.
- Network Data: Gather information about spectrum usage and signal quality as well as about the interference.
- Modulation and Channel Conditions: Store information regarding one or more modulation techniques and channel characteristics.

2. Data Integration:

- Combine Data Sources: Information such as user information, network information, as well as channel conditions should be integrated into one dataset.
- Format Data: Make sure all the entries in are of a similar format and use the same measurement units where applicable.

3. Data Preparation:

- Feature Extraction: Some of such features may include user demands and their patterns, signal-to-noise ratios etc.
- Data Labelling: Provide labels according to characteristics of the signal and the need of user.

4. Data Splitting:

- Split Data: Further split the data collected or gathered into three sets namely training set, validation set, and test set.
- Ensure Balanced Data: Ensure that each subset is equitably peace and parallel.

5. Data Preprocessing:

- Normalize Data: After this, standardize the features present so that you will find that they are alike.
- Handle Missing Values: Check and update all the records, which contain missing or incomplete data entries.

3.3.Machine Learning Model Building

Objective: To Design and implement an enhanced deep neural network model that enhances spectrum utilization for Hybrid NOMA networks.

1. ANN Multi-User Detector (MUD):

   Network Architecture: Develop the artificial neural network (ANN) with the input layer, hidden layer and output layer. In hidden layers apply ReLU activation functions and apply suitable activation functions in the output layer (e.g. sigmoid or SoftMax).

2. Training Algorithm:

   Levenberg-Marquardt Optimization Algorithm (trainlm): Overlapping, idle for medium size of the network, it is used to adjust the weights of the neural network to reduce the mean squared error.

3. Data Split: Select the 70%, 15%, and 15% of the entire data set randomly to split the data into training, validation, and test sets for a more proper training and evaluating.

4. Performance Measurement:

   Mean Squared Error (MSE): Employed in the identification of average squared deviation of predictions from actual value in determining the performance of the model.

5. Computation Efficiency:

   MEX (MATLAB Executable): Make use of MEX to enable the computations to go through faster to arrive at the training algorithm.

3.4. System Architecture Design

Goals: Combining the ANN model in a NOMA network system and proposing an efficient architecture for this new approach.

1. System Integration:

    - This either involved the integration of an ANN MUD within existing infrastructure in NOMA network or a deployment strategy to make it compatible with current architecture.
    - DSP (Digital Signal Processing) in the State of Dynamic: Propose dynamic scheduling algorithm to dynamically allocate spectrum based on real-time or users 'desired demand.

2. Simulation and Testing:

    - Implementation in MATLAB: Create the ANN MUD within a MATLAB code and then test it by simulating different modulation schemes 8PSK modulated messages, Phase Shift Keying (PSK) various modulated messages, and Quadrature Amplitude Modulation (QAM) modulated messages and some MIMO configurations.
    - Performance Metrics: Measure increases in throughput, spectral efficiency and other KPIs (Key Performance Indicators) benefits to users.

3.5. Performance Assessment

The flow chart in Figure 3.1 depicts the steps that will be followed when optimizing the spectrum allocation in a Hybrid NOMA network. This process is divided into several stages, each of which is described in detail below. By following this process, it is expected that the spectrum usage utilization of hybrid NOMA networks can be significantly improved.

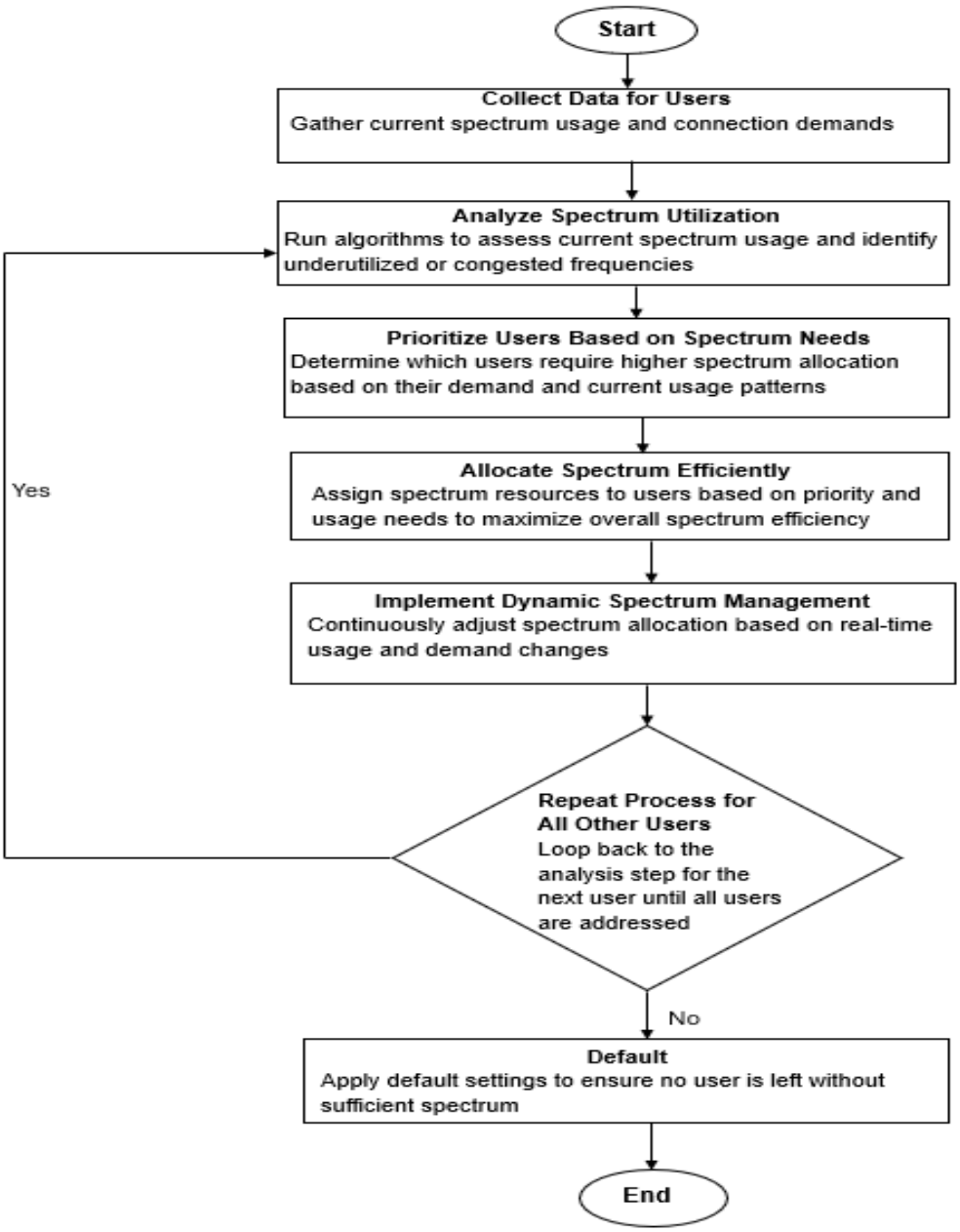


Figure 3.1: Steps to optimize spectrum utilization

The flow chart figure 3.1 for dynamic spectrum management would systematically address the steps for optimizing the spectrum allocation in a Hybrid NOMA network. This will be of paramount importance in enhancing both the efficiency of the network and the quality of the users' experience by ensuring that spectrum resources are utilized well in real time. The process begins with information regarding the current spectrum usage, and their demands for connectivity. It is important because one can gain an understanding of the users' needs and the condition of the network.

Following data collection, the workflow proceeds to the analysis of spectrum utilization. By using algorithms that calculate the usage at that instant, the underutilized and congested frequencies can be identified. The analysis then prioritizes the users in the light of their needs from the spectrum, determining which user should have more spectral resources.

The core of the workflow is the efficient spectrum allocation, whereby the resources are granted to users based on priority and usage patterns to maximize overall spectrum efficiency. Dynamic management of the spectrum makes use of constant optimization that is capable of real-time changes in optimizing the allocation based on changing usage conditions and demands.

This is an iterative process, repeating a loop back to the analysis step for each of the users until all are accommodated. Also, default settings are applied, or edge cases are handled, so no user would remain with poor spectrum. The developed workflow aims at improving network performance by balancing user demands with available spectrum, thereby optimizing efficiency and the quality of service in the NOMA network.

CHAPTER FOUR: SYSTEM DESIGN

This chapter elaborates how the system adopted to manage spectrum usage to improve the hybrid NOMA networks operated through a complex Machine Engineering. This chapter categorically describes and analyses the strategies and techniques of enhancing the framework of the spectrum efficiency particularly the deep neural network (DNN) based MUD approach for implementation and assessment. It embraces virtually all issues including the neural network design employed in the creation of the design, and the algorithms and techniques used in managing the spectrum. This chapter is divided into three sections:

1. **System integration**

DNN system: In this part, the basic structure of the application of the Deep Neural Network (DNN) in the proposed method is presented. It encompasses information of the layers and nodes in the network, as well as the activation functions used to facilitate learning through a DNN about the input data and enable the prediction of the likely outcome. The proposed method presents the new method called Multi-User Detection Neural Network (MUD_NN) which is developed for MUD task in NOMA networks. It solves the problem of efficient utilization of spectrum through the neural networks capacity of differentiating and filtering different signals from different users. The section also explains how MUD_NN can bring improvement to the efficiency of the spectrum and increased performance of the network. Additionally, the application of Singular Value Decomposition (SVD) is used with emphasis on it in relation to the design of the system application. This is done to improve the learning ability and performance of the neural network, through applying SVD on the training input data to identify features which are pertinent to training.

2. **Workstation**

This chapter deals with the technical specifications and rationale for choosing the workstation that was being used in this study. The laptop, a Lenovo computer with an Intel® Core™ i5-10210U Central Processing Unit (CPU) and 16 GB of Random Access Memory (RAM), was picked because it is a budget type, and it can work well for the purpose of the computing work that is required in the laboratory.

3. **Algorithm implementation**

This section discusses the process of applying the machine learning algorithm suggested in the text in MATLAB. It explains how the proposed method, Deep Neural Network Multi-user Detector (DNN_MUD), is used for various modulation types like 8PSK (8-Phase Shift Keying), PSK (Phase Shift Keying) and QAM (Quadrature Amplitude Modulation). In addition, it

discusses application of different multiple input and multiple output (MIMO) configurations to study the capability and efficacy of the algorithm in numerous conditions.

4.1. System integration

DNN system

The elemental structure of a Deep Neural Network (DNN) is organized into three primary components: input layers, hidden layers, and output layers. The input layers serve as the initial point of interaction, getting information within the shape of networks or tensors from outside sources such as clients or sensors. These layers act as conduits, transmitting the input information forward to the ensuing layers of the arrange without performing any computational changes themselves. The relations between the input layer, hidden layer, and output layer showing the structure of a deep neural network with one hidden layer is presented in Figure 4.1.

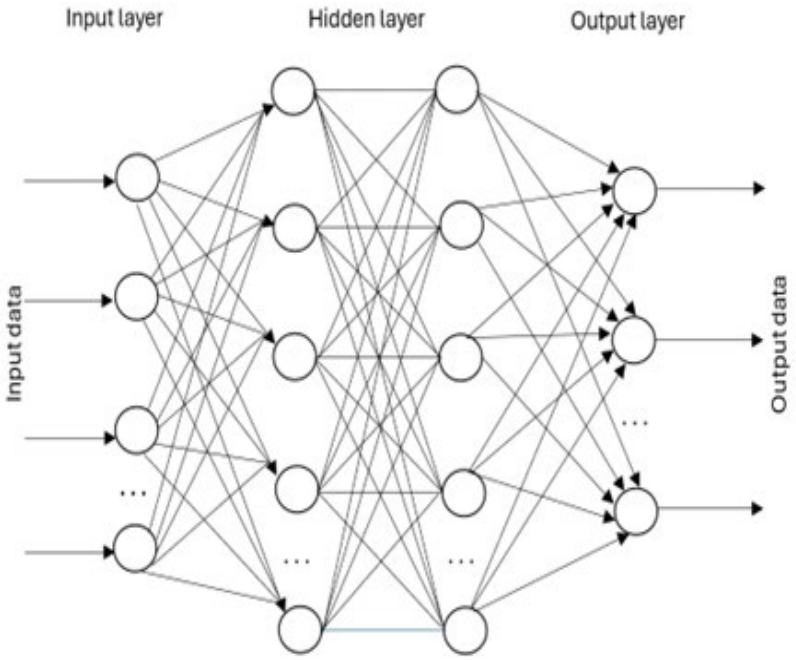


Figure 4.1: Structure of a deep neural network (DNN)

The hidden layers constitute the centre handling units of the DNN. Comprising different layers of interconnected neurons, these hidden layers are dependable for extracting and refining features from the input information. Through the arrangement of weighted associations and activation capacities, each hidden layer dynamically learns to abstract higher-level representations from the crude input. For example, in picture recognition tasks, lower hidden layers might perceive essential features like edges and surfaces, whereas deeper layers

recognize more complex designs such as shapes or objects. This progressive include extraction capability is key to the DNN's capacity to show complex connections inside information.

The output layers get the processed data from the final hidden layer and deliver the ultimate outputs of the network. The nature of these outputs' shifts depending on the assignment the DNN is outlined to perform ranging from course probabilities in classification tasks to numerical forecasts in relapse errands. Activation capacities tailored to the output requirements, such as SoftMax for multi-class classification or straight actuation for relapse, are regularly connected in these layers to create significant forecasts or choices based on the learned features.

Generally, the organized stream of information through input, hidden, and output layers shapes the foundational design of a DNN. This architecture empowers the model to independently learn and show complex designs and connections inside information, making DNNs exceedingly flexible and compelling over differing spaces such as picture recognition, natural language processing, and beyond.

Proposed methods

The proposed method MUD_NN (Multi-User Detection Neural Network) for NOMA addresses the challenge of ensuring an efficient use of the spectrum. In the earlier orthogonal multiple access (OMA) techniques like time division multiple access (TDMA), and frequency division multiple access (FDMA) every user is assigned a different time slot and frequency band, respectively. Hence, the available spectrum is not utilized optimally and becomes a problem where the user's demand and traffic distribution are varying.

Non-orthogonal multiple access (NOMA) is as a major shift from the traditional concepts. It grants multiple users the ability to use the same time-frequency resources at the same time. This enhances the spectral efficiency using power domain multiplexing, which distinguishes the users in terms of power and orthogonal time or frequency. Nonetheless, interference management, and user detection are some of the challenges arising in NOMA because of overlapping curves at the receiving end. This work presents a neural network based multi-user detector (MUD_NN) for NOMA as follows to motivate the design: Enhancing the receiver's capability to demodulate the signals from multiple users in NOMA systems. It is evident that traditional receivers fail to distinctly demodulate and decode concurrent signals as they experience severe impairments in distinguishing between the interfering signal and the wanted signal with deteriorated channel conditions and interference level.

MUD_NN employs a deep neural network towards the enhancement of MUD in the NOMA system. It is worth to underline that MUD_NN for NOMA works in the context of specific system model that is designed in accordance with the tenets of NOMA. In this model:

- User multiplexing: enables many users to try to connect to the same time frequency resource with different power levels.
- Superposition coding: At the transmitter side, the signals received from all the users are added up by employing the use of super positioning coding methods.
- Receiver complexity: On the receiver side, the MUD_NN employs a deep neural network to accurately detect as well as decode signals originating from multiple users even in cases where the power of a signal is in the same range as the other users' signals.
- Learning and adapting: MUD_NN's neural network as a result learn power levels and channel conditions for picking up distinct signals and having adjusted ability to modify and enhance the signals for different networks and achieve higher accuracy of signal detection.

The MUD_NN integrates deep learning techniques into the NOMA framework to improve the receiver's ability to decode signals from multiple users while minimizing interference and improving spectrum usage efficiency. This approach not only increases the overall capacity of wireless networks, but also contributes to enabling next-generation communication systems that can support diverse and demanding applications such as high-speed data transmission, large-scale IoT connections, and ultra-reliable low-latency communications (URLLC). As research in this field continues to evolve, the MUD_NN represents a promising direction for NOMA in optimizing spectrum utilization and enhancing the capabilities of wireless communication technologies.

Singular Value Decomposition (SVD)

a) Role of Singular Value Decomposition (SVD)

Role of Singular Value Decomposition (SVD) on the processing of the Training Input

Singular Value Decomposition (SVD) was performed on the training input, which in this case is a multiplexed modulated message encoded with 8PSK (8-phase shift keying). The main role of the SVD operation in this context is to execute the processes of:

1. Characterizing the Training Input: The SVD operation decomposes the training input matrix into three matrices: U, S, and V. The matrix S contains singular values that are unique and help to represent the features of the training input. Each training input

(multiplexed modulated messages) has its own set of singular values that contain important, and unique information about the data structure.

2. Feature Extraction: By extracting the singular values from the S matrix, an accurate representation of the training input is obtained. These singular values function as unique features that can be used for further analysis or as inputs to machine learning models.

b) Justification for using SVD

- Uniqueness of Singular Values: The singular values derived from the SVD are unique for each input matrix. This allows each multiplexed modulated message to be uniquely characterized, facilitating identification and analysis of the training inputs.
- Dimensionality Reduction: SVD provides a way to reduce the dimensionality of high dimensional data. By focusing on significant individual values, the process preserves important information and discards redundant or less important data, simplifying further analysis and computation.
- Noise Reduction: SVD effectively separates noise from signal. Larger individual values represent essential features of the data, while smaller values often correspond to noise. Focusing on larger individual values removes noise, improving the quality of the data representation.

c) The number of singular values resulting from running SVD.

The number of singular values resulting from running the SVD process is equal to the minimum number of rows and columns of the input matrix. For the multiplexed_message_8PSK matrix of dimensions is 30 x 50, there are 30 singular values they carry several key properties of this matrix. They describe the rank, and the number of linearly independent rows of matrix, that furnish effective dimensionality of data. Each singular value represents the variances contained in a particular principal component. The higher the singular value of a component, the more variance within the data that it contains. Since there are 30 singular values-that's the number of rows in this matrix-the matrix is full row rank, meaning there's no more redundancy than can be described by these 30 components. These singular values are most critical to the reconstruction and compression of matrices since they retain key features and structure of the original data.

d) Entities that lead to the creation of the training input

- Modulation scheme: The use of 8-phase shift keying (8PSK) modulation to encode the message.

- Multiplexing: The process of combining multiple 8PSK modulated messages into a single matrix, thereby generating the training input.
- Training Input Matrix: The resulting matrix (multiplexed_message_8PSK) of dimension 30 x 50 represents the combined and multiplexed modulated messages.

e) System entities that lead to the creation of the training output

- SVD Components: The training output SVD components reduces the matrix A into three different matrices: U, S, and V. Each of these matrices has some role to play in representing the initial matrix into a more manipulative form:
    - U (Left Singular Vectors): U is an orthogonal matrix whose columns are the left singular vectors of the initial matrix A. Each column of U represents a direction in the input space along which the data varies. In other words, these vectors form an orthonormal basis for the column space of the matrix A. If A is an **m x n** matrix, then U will be an **m x m** matrix. The columns of U provide the basis vectors for the row space of A.
    - S (Singular Values): S is a diagonal matrix containing the singular values of A; it carries the non-negative numbers describing the magnitude or strength of each corresponding singular vector. They are ordered such that the first few singular values grab the most important features of the data, and the smaller ones correspond to less important features. The matrix S has dimensions **m x n**; the singular values are on the diagonal, while all other entries are zero.
    - V (Right Singular Vectors): V is an orthogonal matrix whose rows are the right singular vectors of the original matrix A. Each row of V gives a direction (in the output space) along which the data projects on. If A is a **m x n** matrix, then V will be an **n x n** matrix. The rows of V give the basis vectors for the column space of A.
- Singular Values: The diagonal elements of the S matrix, which represent the singular values, are used as the training outputs. These values capture the essence of the training inputs and act as a reduced representation for further processing and model training.

In summary, the SVD operation is crucial to effectively extract a unique, reduced, noise-filtered representation of the training input in terms of its singular values, enabling efficient and accurate data characterization.

4.2. Workstation

Technical configuration of the workstation

In the case of our research, we utilized a Lenovo laptop fitted with an Intel® Core™ i5-10210U processor engine. On the one hand, the CPU has a base frequency of 1.60 GHz and can reach up to 2.10 GHz thus offering a good trade-off between power and energy consumption that is important for computing' (Lenovo, 2024). The notebook was equipped with 16 GB of RAM which is enough to store a lot of bytes and a suitable tool for scenarios where there are significant constraints. The 64-bit operating system is the key requirement to ensure that all the features of the x64-processor architecture that the system uses are utilized' (Lenovo, 2024). Such a setup enables the laptop to deal with sequential operations effectively as well as perform the basic calculations that go into defining and testing machine learning models' (Lenovo, 2024).

Main factors for choosing a laptop workstation

The main driving factor was the initial practicality and the affordability that the device provides. This laptop cost about R21,529.38 which is not much considering that we save much more if we don't spend money on using very expensive servers or specialized hardware (Lenovo, 2024). This choice not only assists in the successful accomplishment of research goals by efficiently managing expenses but also results in the greatest computational capacity possible. Thus, the need to apply such a laptop arises due to the absence of appropriate tools resulting from limitations in design. In some settings, it might be very difficult to avail high-performance computing equipment, especially when the setting's budget is extremely tight. Therefore, we came up with a solution that could be used to create and run Multi-User Detection Neural Network (MUD_NN) on a workstation that is user-friendly and cost-effective. Although it is less costly, the configuration of the laptop with a reliable processor and enough RAM enables data management and machine learning without the purchase of high-cost computational resources' (Vivienne Sze, et al., 2017).

An Intel Core i5 CPU and 16 GB of RAM are the technological characteristics of the Lenovo laptop which make it suitable for applications such as Singular Value Decomposition (SVD) and neural network training (Thi, 2023). A laptop is our data preparation and feature extraction tool; therefore, we would not be able to complete our jobs without its fantastic performance. In fact, Thi (2023) says that the laptop's features can carry out these two tasks efficiently. The addition of a high-performance CPU and ample RAM to CPU has a large effect on our tasks

because with the laptop we can provide performance levels that are required in our research tasks' (Thi, 2023).

4.3. Algorithm implementation

These results will be achieved using a machine learning algorithm developed in MATLAB. The algorithm is scalable and able to manage large amounts of data. It is robust and accurate, even in the presence of noisy data. The algorithm is computationally efficient, generalized to be applicable to other types of networks and applications. Machine learning algorithms are effective tools to improve network performance even in complex and dynamic environments. The algorithm developed in this research has potential applications in many fields, including internet of things, and mobile networks. The main goal of the proposed method was to investigate the performance of DNN_MUD algorithms for 8PSK modulated messages, Phase Shift Keying (PSK) various modulated messages, and Quadrature Amplitude Modulation (QAM) modulated messages using multiple input and multiple output (MIMO) topologies. This section will discuss the parameters used in Deep neural network multi-user detector (DNN_MUD) design.

Deep neural network multi-user detector (DNN_MUD) parameters

Table 4.1 lists the key parameters of the proposed DNN_MUD design. Each parameter is crucial to understand how the network engages in multi-user detection within a Non-Orthogonal Multiple Access (NOMA) environment. The table shows a general outline of the DNN settings, training input, input size, code length, user count, message length, and number of layers.

Table 4.1: DNN_MUD structure

Layer	Parameters
Training input	8PSK
Input size	60
Number of inputs	1
Code length	5
Number of users	6
Message length	50
Number of layers	3

- **Training input:**

The DNN MUD training input uses 8PSK (8-Phase Shift Keying) modulation. 8PSK was found to be more economical in data transmission with the increase of the length of phases than the simpler modulation. This choice affects the complexity of the data that the network must be

trained to process, which is crucial for successful multi-user detection. As compared to other simpler methods of modulation, the 8-PSK modulation is more complex, but in many cases, it is less costly in data transmission because the transmitting of a greater number of information can be done with a single symbol. This is so because scientific management has several factors that make it efficient:

- Other benefits include Higher Data Rate per Symbol: In 8PSK one symbol transmits 3 bits because 8PSK signal has 8 phase states and it can represent $2^3 = 8$ symbols. This is much better than other forms of modulations including Binary PSK (BPSK) which only modulates 1bit per symbol and Quadrature PSK (QPSK) which modulates 2 bits per symbol. In other words, for the same time, 8PSK can deliver more data; therefore, it will be more suitable for high data rate applications' (Vineela & Madhu , 2018).
- Spectral Efficiency: Spectral efficiency is defined as a ratio defined by the ability to transmit data through a given bandwidth. 8PSK stands for 8 Phase Shift Keying and provides a higher spectral efficiency as it has the capability of transmitting 3 bits per symbol as compared with BPSK and QPSK. This makes it possible to achieve higher data rates under the same bandwidth as will be discussed below' (Yue , et al., 2020).
- Trade-offs in SNR: 8PSK has more date rates than QPSK; however, operates are difficult since the phase states are closer and thus a high signal to noise ratio is needed for effective transmission. This trade-off is relevant in the determination of effectiveness of the modulation scheme given the channel status at present. However, considering the utterances made above, this is if the signal to noise ratio is high. The benefits for 8PSK are the problems that are associated with high sensitivity to noise' (Kaiyuan , et al., 2020).
- High data-rate application's comparative performances: The 8PSK system is, in some sense better than QPSK in the scenarios where information must be transferred at a higher data rate without the expansion of the bandwidth; this is achieved through the process of transferring more data in the same amount of bandwidth. This is useful when the bandwidth is minimum, and this is well on its way to being 'the norm' in today's environment where information on the internet is in abundance' (Xuan , et al., 2023).

The 8PSK methods of data transmission therefore achieve efficiency specifically due to the over proportional number of bits per symbol together with high spectral efficiency. Although it employs higher Signal-to-Noise Ratio (SNR) transmission of the signals, it enhances the functionality of the bands as well as that of the data rate; therefore, it should be eligible to be employed in the modern communication systems.

- **Input Size:**

The input size refers to the dimension of the feature vector fed into the artificial neural network. This dimension is obtained from pre-processed data containing signal characteristics and any relevant features extracted from 8PSK modulated signals. The bigger input size allows the network to gather more specific information from the signal, hence it is vital for the identification of many users in a NOMA system.

- **Number of inputs:**

The number of inputs for this problem is synonymous to a singular vector that contains a numerical description of the entire dataset. This represents the vectorized version of 8PSK signal data that the network uses to detect multiple users.

- **Code length:**

The code length refers to the length of the codes used in the modulation technique. In the case of 8PSK, this might be a number that shows the symbol per bit, or the length of the data sequences used for training. If the code is 5, it means that a five-bit code is being used to transmit the signal, which will have a direct impact on the neural network's structure and the capability of the network to model the relationships between different codewords and user inputs.

- **Number of users:**

The number of users is a number that tells us how many unique user signals must be distinguished and detected by the DNN MUD in the system. This link influences the network's effectiveness in multi-user scenarios, and it also makes sure that the model is trained to manage and separate signals from up to six different users simultaneously.

- **Message length:**

Users' messages reflect the length of each message. A 50-bit message regulated and coded states that each message length is 50 bits long. The value of this parameter shows the size of the data that must be processed and stored by a network, affecting the network's capability to cope with the length of messages sent by different users.

- **Number of layers**

The DNN MUD structure has a 3-layer configuration. This is made up of one input layer, one or more hidden layer, and one output layer. The number of hidden layers arise from the depth of the network which is an essential requirement for the collection of complex patterns and correlations in data. A three-layer network is the outcome of compromise and is both computationally practical as well as the relevant complexity to describe the problem of multi-user detection.

4.3.1. Algorithm analysis

In this section, the process of realizing multi-user communication system involving NOMA, modulation techniques and application of DNN are discussed. This implementation mimics messaging of users and spreading their messages, uses modulation techniques and tests different types of communication based on the Multiple-Input Multiple-Output (MIMO) channel model. The code explicates a complete way of tackling on signal processing and machine learning even in the field of wireless communication systems.

The section includes several key components:

Spreading of Messages: The code generates random binary messages and spread codes for every user and subsequently applies the codes obtained for the spreading of the messages as a way of practicing NOMA.

Modulation: Various modulated techniques including 8-PSK and QAM are some of the modulation methods used to condition the messages for transmission as a signal.

Channel Simulation: The MIMO channel model is employed to emulate the action of multiple transmitters as well as multiple receivers on the modulated messages.

Neural Network Training: Multi-user detection process is carried out with the aid of a deep neural network on the received signals with an objective of enhancing detection accuracy.

Most of the code is included in the Appendices section.

1. **Initialization and Message Spreading**

It also begins with **'clear all'** and **'clc'** which are MATLAB commands to wipe out all the variables in the working environment and erase all the texts that has been typed in the command window respectively. This is important to make sure that the working environment is clear before the code can be run.

Parameters Definition:

- **'codeLength'** defines the spreading codes' length.
- **'numUsers'** describes the number of users in the system.
- **'messageLength'** determines the number of characters a user can input in a message.

Generating Spreading Codes and Messages:

- **'spreadingCodes = randi ([0,1], codeLength, numUsers);'** generates matrix of random binary values (0's and 1's) and where each of the vectors represents a user's spreading code.

- **'messages = [0, 1; randi (messageLength, 1, numUsers);];'** randomly generates the binary messages of length message length for the number of users.

The code then extracts individual user codes and messages for processing:

- **'User_1codes, User_2codes'**, etc., contain the spreading codes assigned to different users.
- **'User_1mes, User_2mes'**, etc., Store the messages of each user in a format that is transposed for different dimensions for further processing.

Spreading of Messages:

The spreading process is the multiplication of each of the users' codes by the message:

- The nested **'for'** loops transverse the spreading codes and messages ending up into the signal space.
- **'User_1spm (i, j) = User_1codes(i). * User_1mes(j);'** converts the message to multiply each element of the user's code to spread out the messages accordingly.

2. Modulation

Modulation is perhaps one of the most important of these processes since it entails the conversion of spread messages into modulated signals. The code supports different modulation schemes:

- Profile 1 - 8-PSK Modulation:

- 8-PSK (8 Phase Shift Keying) is a type of modulation technique in which each symbol of modulation is represented by one of the eight phase shifts of the carrier wave. This approach encodes 3 bits per symbol (since 2^3=8) and is intended to provide good bandwidth application while providing moderate noise tolerance.
- Complexity: Provides compromise between the overall bandwidth popularity and the architecture system.
- In this process the **'pskmod'** function is employed for modulation of the spread messages into 8-PSK (Phase Shift Keying) format. The parameter **'pi/4'** define the phase shift in the output x(t) signal.
- **'Multiplexed_message_8PSK'** involve the combination of modulated signals belonging to all the users.

- Profile 2- Various PSK Modulation:

Different PSK refers to a few of Phase Shift Keying techniques which can be any of 8-PSK or other types. This profile builds upon the Profile 1 by implementing various PSK schemes

namely 8-PSK and 16-PSK. All the variants raise the number of phases states and hence serves to extend the number of bits that can be transmitted per symbol.

- In this profile, different modulation schemes are applied to different users:
  - Only the users 1, user 2 and user 4 employs 8-PSK modulation.
  - Users 3 and 5 use 16-PSK modulation.
- Higher Data Rates: For instance, through higher order PSK schemes such as 16-PSK there is the additional advantage of encoding more bits per symbol than in 8-PSK thus leading to an increased data rate.
- Flexibility: Offers flexibility in the modelling and simulation of phase modulation schemes which can be characterized by diverse features such as data rate and error performance.
- Adaptability: Facilitates in ascertaining how different PSK schemes work under different channel environments and amount of noise.
- With reference to the preceding, the **'multiplexed_message_PSK_various'** synthesizes the messages using different PSK techniques.

➢ Profile 3 - QAM Modulation:

Quadrature Amplitude Modulation (QAM): QAM further extends profile 2 by incorporating the modulation techniques involving both the amplitude and phase shift. While in PSK it modulates only the phase of the signal. QAM modulates both the amplitude and phase of the signal, which means it allows a greater number of symbols and hence much higher data rate.

- Greater diversity in the number of symbols: QAM can represent more symbols than PSK because it makes use of different possible values of amplitude shift in addition to phase shift. For instance, 16-QAM means that there are 4 bits transmitted per symbol since $2^4 = 16$.
- Improved Scenarios Modelling: These features allow the modelling of communication scenarios where variations in both amplitude and phase are significant, thus providing a better view of signal transmission.
- The next block of users uses **'qammod'** function for QAM modulation.
- '**Multiplexed_message_QAM'** – contains QAM modulated messages.

3. MIMO Channel Simulation

The **'comm. MIMOChannel'** is used to simulate a MIMO channel, and other phases include Interference, Coding, and Modulation. The MIMOChannel Object defines a multiple input and multiple output path wireless channel, used to analyse the high-end wireless communication systems which makes use of multiple transmitting and receiving antenna to improve data over targeted distance. Parameters include:

- The channel characteristics are **'SampleRate'**, **'PathDelays'**, and **'AveragePathGains'** as influenced by the considered channel model.
- There are two factors that belong to the control of the channel: **'MaximumDopplerShift'** and **'SpatialCorrelationSpecification'** which determine the Doppler effect and spatial correlation of the channel in the analysis separately.
- The simulated transmission of multiplexed modulated messages through the MIMO channel, produce received messages for each modulation profile:
    - Receieved_Message_1, Receieved_Message_2, and Receieved_Message_3 are the signals which were received for different modulation schemes.

4. Neural Networks Configuration and Training

Neural Network Setup:

- By using **'network'** object a deep neural network (DNN) is developed. It comprises one input layer, one hidden layer and one output layer.
- The **'transferFcn'** sets the activation functions for each layer:
    - This also applies to the input and hidden layer employing 'tansig' – tangent sigmoid activation function.
    - The output layer uses 'logsig' (logistic sigmoid) function in the present problem.

Training Data Preparation:

- The received messages, which are **'Receieved_Message_1'**, **'Receieved_Message_2'** and **'Receieved_Message _3'** are used to train the preparations.
- **'Training_Input_1'** and **'Training_Output_1'** are reshaped as well as normalized. Training data is divided to groups and normalized to provide better results for the neural network.

Training the Neural Network:

- **'trainingOptions'** sets factors governing the training of a model, some of these options include the optimizer type where we set it as **'adam'**, the maximum number of epochs

permissible, mini-batch size that specifies the number of training examples given to the neural network at a time and verbosity level.

- To explain the improvement that is achieved through mini batch training the script explains how the data is shuffled and segmented into batches.
- **'train'** function is used for training the artificial neural network with the help of the prepared data whereby it determines the performance metrics such as MSE.

SVD Computation:

- Singular Value Decomposition (SVD) is executed on the signals of **'8-PSK Modulation'** to identify its characteristics.
- Eigenvalues are computed and scaled for them to be suitable for feeding training data in the neural network for training.

5. Results and Visualization

Displaying Results:

- The program uses **'disp'** to print various results:
  - Created signature and random codes to be spread.
  - Multiplexed message matrices.
  - SVD computation results and the dimensions that where they belonged.
  - Training inputs and outputs.

Training Progress Visualization:

- The **'plot'** function shows the mean squared error describing the artificial neural network performance for the training, validation, and testing phases in epochs' scale. They include visualization that assist in evaluating the learning performance of the neural network above.
- Additional **'view**' function enables a viewing of the network structure.

Final Configuration:

- The is the determination of training parameters such as the training epochs, learning rate as well as the overall performance function.
- The **'train'** function is simply called with new parameters linking to fitting the model.

Finally, this code generates a higher-level analogy of a real communication model incorporating several phases of signal transmission, modulation, channel emulation, and ANN training, and yields the multi-user detection efficiency in MIMO communication.

CHAPTER FIVE: SIMULATION AND EXPERIMENTATION

The discussion in this section will help to get into the processes of modelling and testing. In the beginning, the spreading codes are generated, which encrypt user messages during the transmission. In other words, these codes are at the heart of the systems in NOMA. They spread the messages of the users over several frequencies such that effective signal processing is possible. Initially, random binary messages are generated and then, they are multiplexed employing 8PSK modulation, giving rise to a steady signal stream transferred to a Multiple-Input Multiple-Output (MIMO) channel. This way, the system is put in place to be able to diagnose the situation and see its behaviour under practical scenarios. Singular Value Decomposition (SVD) is applied on these multiplexed signals to see how they are composed and to gain the singular values, which can be used to tell the signal strength and system behaviour.

The output of the neural network is then normalised and quantised so that it can meet the learning requirement of the network. The feed-forward neural network being used in relation to the MUD is comprised of specialized layers and activation functions, which in turn enable it to successfully manage various user signals. Trainer teachers are individuals responsible for setting up the right parameters and optimization techniques to achieve specific learning and detention. The chapter concludes with the evaluation of the training results, which involves several indicators like Mean Squared Error (MSE) and improvement through progress visualization during training. These explorations give a clear understanding of the network's capability to recognize and to differentiate multiple user signals within the NOMA paradigm.

5.1. Generated Spreading Codes

The line of code produces a matrix called Spreading Codes with dimension code Length as number of Users as shown in Table 5.1. Each element is randomly a 0 or 1. The result shows a 5x6 matrix where each column represents group of bits of the spreading code. The command generates a spreading code for a communication system where multiple users transmit data simultaneously on the same channel, in a spread spectrum communication system The randomness of the generated codes minimizes the dependency of one code on the other, and thus helps in the separation of the signals at the receiver. Certain values of 0's and 1's in the generated code can impact system performance, including factors such as interference mitigation and error correction features**.**

Table 5.1: Random Generated Spreading Codes

Generated Spreading Codes					
1	0	0	0	1	1
1	0	1	0	0	1
0	1	1	1	1	0
1	1	0	1	1	1
1	1	1	1	1	0

5.2. Generated Random Messages

Results shows generated matrix of 52x6 called messages with dimensions messageLength by numUsers, where each element is randomly chosen to be either a 0 or 1. Each row represents a message to the user. Each column represents a bit in the message. Inference: These randomly generated messages represent data or information that each user wants to send through the communication system. Randomness ensures that messages are different and is useful for testing the robustness of communication systems to different types of data. Certain patterns of 0's and 1's in messages can affect system performance, especially regarding error detection and correction.

5.3. Multiplexed message 8PSK.

The results show a complex-valued matrix of dimension 30x50. Where 30 is the length of the multiplexed message. Each column of this matrix represents a different user's modulation message. The complex numbers in the matrix represent the phase of the modulating signal.

Inference: Multiplexing these messages allows multiple users to send data simultaneously on the same channel, increasing the throughput of the entire communications system. Uses 8PSK modulation, in binary terms 8 is 2^3. This means that each symbol represents 3 bits of data (8 phase shifts), resulting in higher spectral efficiency compared to simpler modulation schemes such as BPSK (Binary Phase Shift Keying) and QPSK (Quadrature Phase Shift Keying), which represent 1 and 2 bits per symbol, respectively.

5.4. Singular_values_8PSK

It is established that the Singular Values of 8PSK are what is meant here. These are the vital elements of the matrix $S8PSK$, which is formed by the SVD surface representation of the 8PSK modulated message matrix. The SVD is a method that separates the matrix into the three basic parts: U, S, and V. S stands for the matrix of the singular values which is of immense importance when it comes to data analysis and interpretation. This matrix is noted Σ in mathematical form and it is diagonal where each entry is a singular value that determines the significance of each of the principal components in the given data set. Here the diagonal of the matrix S, provides the measure of variance explained by each of the components thus helping to identify the extent to which these components generalize the underlying structure in the data. These are high values where each feature retains job on providing remarkable differentiation within the data set to be analysed; low or zero values imply a condition of redundancy of some features in the data set due to existence of linear relations among them. The relevance of S lies in the simple view on the data which can be provided with its help, as well as the possibility of dimensionality reduction and efficient data compression. One major aspect of a dataset is that several features have less importance in relationship to other features, consequently, with the help of the biggest singular values, only the most important features are selected. It increases the result and decreases the complexity of calculations.

The singular values present in $S8PSK$ are indicators of the major part magnitudes of the 8PSK signal matrix. They are sorted in a descending order, which causes their relevance to be precisely represented in a ranked list or "components list". Higher singular values are components that have a larger fluctuation and are therefore important to capture, whereas lower singular values have lesser fluctuations. Singular values that are less than one may have negative values, which indicates noise. The investigation of these singular values allows for the analysis of the signal's structure and its ability to faithfully represent the signal.

Analysing Table 5.2 is critical as it consists of information about the dataset and its characteristics as well as its processing. The values that are given in the table show to what degree each of the first principal components affects the total variance of the dataset. Its non-zero singular values indicate the primary directions of variation that reflects the major patterns and characteristics of the given data. The presence of a singular value of zero suggests that S_8PSK has less than 30 linearly independent components, meaning redundancy and that some

features are linearly associated and therefore the dataset is not informative. This understanding is vital for the process of dimensionality reduction as, it makes it possible to omit some unimportant features while retaining the important information which helps to ease the data processing operation and enhance computational efficiency. In other words, by studying these unique values, it becomes easy to filter the dataset down in a way that makes the rest of the work on the data much easier.

Table 5.2: Singular values 8PSK

Singular values
37.4816
5.9922
4.9477
4.4113
2.9971
2.5106
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000

5.5. Training_Input_1

The code first pre-processes training input data by defining and then manipulating several variables that play a key role in preparing data for machine learning model training. Variable names and their significance are explained herein.

Multiplexed_message_8PSK: Variable carrying the message data modulated in 8PSK format. It is information carried by several users modulated 8PSK signal. By setting the variable of multiplexed_message_8PSK to Training_Input_1, the modulated signal data is used from raw modulated signal data to prepare the training input data.

Training_Input_1a_grp_1 to Training_Input_1a_grp_13: These are variables for elements of the real component of Training_Input_1. Data is segmented into different categories for further and deeper processing and analysis. This segregation allows for deeper processing and analysis that will help in debugging, feature creation, and modifications for the skew data.

Training_Input_1: Following the same process, after the segmentation and processing has been done, Training_Input_1 is redefined by joining (horizontally) the groups depicted as Trading_Input_1(a_grp_1 to a_grp_13). This merging compiles all the segmented real parts into one set that is further used towards training the neural network. The first 240 elements have been chosen so that not all data is used for training instead only the most informative and complete samples are represented. This is useful in ensuring that only relevant data are used when the model is being evaluated.

Processing training input data: First, assign the variable multiplexed_message_8PSK to Training_Input_1. Next, an attempt is made to horizontally chain some variables (Training_Input_1a_grp_1 to Training_Input_1a_grp_13) and assign the result to Training_Input_1. In the performance evaluation, the first 240 elements of Training_Input_1 are selected and assigned to the Training_Input_1. Finally, assign the variable singular_values_8PSK to the variable Training_Input_1, and overwriting the previously assigned value. A normalization procedure is done as before.

Inference: This code aims to preserve important information while avoiding overwriting and data loss by selecting the first 240 elements of Training_Input_1 for further processing. This approach ensures that subsequent assignments of singular_values_8PSK do not discard other

important information from Training_Input_1, and we also trying to make the results to be readable and easy to be interpreted.

In Table 5.3, input values are part of a dataset to be used for the training of a machine learning model. These are very important since they act as the basis on which this model is being trained to either identify patterns from, or predict based on, the data which will be presented to it. This table presents a sequence of numerical values, presumably part of a larger dataset. This selection process helps in choosing only relevant data; it ensures data consistency, hence optimizing the performance of the machine learning model.

Table 5.3: Training Input _1

Input values
1.0000
0.1599
0.1320
0.1177
0.0800
0.0670
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000

5.6. Training_Output_1

The variable Training_Output_1 is initially assigned to the variable Received_Message_1. Two separate vectors, Training_Output_1a and Training_Output_1b, are created to store the real and imaginary parts of Training_Output_1, respectively. The real and imaginary parts are vertically connected to form a single matrix, Training_Output_1. The resulting matrix is converted to real values by retaking the real part, transposing the matrix, and reassigning the result to Training_Output_1. Inference: The training output data i.e., Training_Output_1 is processed to extract the real and imaginary parts and then combined into one matrix. The final matrix Training_Output_1 contains real values derived from the real and imaginary parts of the first message received. This processed training output data can be used to train machine learning or signal processing models.

```
7.Training_Output_1:
    0.3902   -0.1070   -0.5271   -0.0159   -0.2984   -0.5640   -0.0025   -0.0949
    0.7483    0.1472   -0.7414   -0.5325    0.1608   -0.3792    0.4470   -0.2514
    0.1881    0.2312   -0.6642   -0.0869   -0.0117   -0.7508    0.3684    0.3145
    0.3387    0.1225   -0.8463   -0.3557    0.0553   -0.5297    0.2664   -0.1826
    0.5000    0.0283   -0.7991   -0.3276    0.2435   -0.5240    0.3926   -0.0671
    0.8145   -0.0193   -1.0051   -0.4328    0.2764   -0.5381    0.3263   -0.2667
    0.1097   -0.0932   -0.7233   -0.4180    0.8504   -0.3214    0.1197   -0.1573
    0.1247   -0.1551   -0.6902   -0.4337    0.8511   -0.3530    0.2308   -0.1230
    0.7591   -0.1850   -0.8728    0.0992   -0.2408   -0.4542    0.0599   -0.3329
    0.8042    0.1992   -0.6953   -0.5413    0.0698   -0.4074    0.3801   -0.2813
    0.7404    0.1410   -0.5334   -0.2313    0.1972   -0.3553    0.0105   -0.4290
   -0.3164    0.3325   -1.1713   -0.2966    0.8097   -0.5574    0.0424    0.0557
    0.0699   -0.2006   -0.7191   -0.4117    0.8733   -0.3854    0.2781   -0.0977
    0.3559   -0.1558   -0.5231   -0.0069   -0.2931   -0.5487    0.0006   -0.0871
    0.5321   -0.1105   -0.8872   -0.6073    0.7932   -0.4105    0.0918   -0.3227
    0.6165    0.1673   -0.7170   -0.3922    0.0796   -0.5383    0.2797   -0.0749
    0.7779   -0.1548   -0.8719    0.0917   -0.2486   -0.4603    0.0580   -0.3358
   -0.1510    0.2579   -1.1203   -0.2735    0.9725   -0.5549    0.1584    0.1444
    0.1738    0.2066   -0.6758   -0.0768    0.0072   -0.7485    0.3835    0.3124
    0.3130   -0.0805   -0.6611   -0.4625    0.3205   -0.6235    0.3601    0.0309
   -0.2258    0.2816   -1.1744   -0.2299    0.9889   -0.5472    0.0813    0.1189
    0.2998    0.0242   -1.0335   -0.5612    0.3011   -0.3501    0.1206   -0.3185
    0.1610    0.1104   -1.0558   -0.6005    0.1407   -0.3348   -0.0150   -0.4476
    0.3436    0.2744   -0.4656    0.0561    0.0661   -0.5759    0.0649   -0.0283
    0.5428    0.0753   -0.7772   -0.3523    0.1931   -0.5299    0.3544   -0.0715
   -0.2398    0.2657   -1.1845   -0.2189    1.0000   -0.5542    0.0879    0.1242
    0.3816   -0.0063   -0.8669   -0.2544    0.3244   -0.4927    0.3643   -0.0834
```

Figure 5.1: Training Output_1

5.7. Feed Forward Neural Network

5.7.1. Feed Forward Neural Network for 2 hidden layers

The configuration of the feedforward neural network for 2 hidden layers is presented in figure 5.2. The feedforward neural network associated parameters are presented as:

Network Architecture

- Input Layer
- Size: 1 neuron
- Description: This layer undertakes the task of taking the input data in this format for convenience and ease of computational processing – single feature.
- Hidden Layer 1
  - Size: 64 neurons
  - Components:
    - Weights (W): A weight matrix of the connections of the input layer to the first hidden layer.
    - Bias (b): An intercept term brought into the linear model regarding the inputs and weights.
    - Activation Function: The activation function used to transform the first hidden layer neurons' outputs. The kind of activation function is mostly always nonlinear in type.
- Hidden Layer 2
- Size: 32 Neurons
- Components:
  - Weights (W): The weight matrix for the connection between the first hidden layer with the second hidden layer.
  - Bias (b): The actual Linear Combinations which were added up with the bias term of the previous layer and the weights.
  - Activation Function: The activation function used in the second last layer of the neurons arrived after the matrix computations done through the two hidden layers.
- Output Layer
- Size: 8 Neurons
- Components:

- Weights (W): It is the weight matrix to the output layer of the network where the second hidden layer and the output layer are the input and output layer of the other order respectively.
- Bias (b): The data bias term incorporated with the linear combination of the previous layer's output and the current layer's weights.
- Activation Function: The function which is implemented on the neurons in the output layer of the neural network. This may be a SoftMax function in classification context as it makes the output probabilities easier to interpret and because the probabilities sum up to 1, determines the class to which the input data belongs to in case of classification problems or a linear activation in case of regression problems.

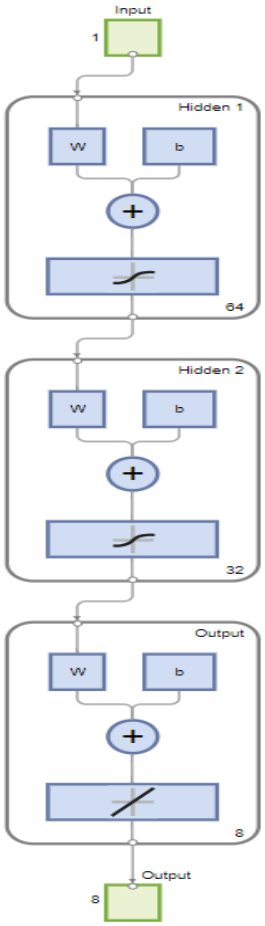


Figure 5.2: Feed Forward Neural Network for 2 hidden layers

Interpreting the Network

- Depth and Complexity: The network has a moderate depth with two hidden layers, allowing it to capture complex patterns in the data. The first hidden layer contains more neurons (64), suggesting that it captures a wider range of features from the input. The second hidden layer has fewer neurons (32), suggesting that more features are extracted.
- Layer Size: Reducing the number of neurons from the first hidden layer to the second hidden layer gradually reduces the complexity of the model, allowing us to focus on the most important features and avoid overfitting.

The FFNN architecture relates the input and output training variables through successive transformation until the inputs get mapped into meaningful predictions. The network starts with a single input neuron that is representative of a particular feature in the network, such as signal strength. Each of these inputs Directed Acyclic Graph (DAG)passes via two hidden layers with a huge number of neurons, using learned weights and biases to model complex relationships in the data. The output from these hidden layers feeds into an output layer containing eight neurons. The nature of the output neurons depends on some task types. For classification problems, SoftMax is used as the activation function in the output layer. It transforms the raw output for each class into probabilities that reflect the likelihood of input to belong to each class. In regression problems a linear activation function is applied in Computational Intelligence in the Data and Network Systems to map the inputs to proceed to the continuous outputs in one step. It thus systematically changes the same training input variable through the layers of the network into output for predictions. This would be in tandem with what the problem requires, whether it be designating classes to data or predicting values in numeric form.

Inference:

- Effective Learning: The network has a reducing MSE during the first epoch, indicating that it effectively learned the underlying patterns in the input data.
- Good Generalization: Similar performance metrics across the training, validation, and test sets indicate that the network does not overfit and generalizes well to unknown data.
- Room for Improvement:

- Additional Layers or Neurons: While the current architecture performs well, experimenting with additional layers or neurons may help capture more complex patterns.
- Different activation functions: Testing different activation functions can improve the network's ability to learn nonlinear relationships.

5.7.2. Feed Forward Neural Network for 3 hidden layers

The configuration of the feedforward neural network for 3 hidden layers is presented. The feedforward neural network associated parameters are presented as:

Network Architecture

- Input Layer:
- Size: 1 neuron
- Description: Specifically, this single neuron input layer oversees receiving the data in the scalar format. From the observation, this layer takes only one feature from the dataset and passes it through to the hidden layers for further process.
- Hidden Layer 1:
- Size: 64 neurons
- Components:
  - Weights (W): An input weights matrix which define the links between the input layer and the first hidden layer. every input neuron is connected to all neurons in this hidden layer, and these are the connection that brings about data transformation.
  - Bias (b): A familiar type of a bias is one that adds an offset whose function is to shift the neurons' outcome of the activation function. This aids in achieving the best outcomes in respect of the performance of the model.
  - Activation Function: Non-linearity, which is applied by means of a non-linear activation function (ReLU, Tanh, or Sigmoid) is introduced in the network so that the network can learn the complex patterns.
- Hidden Layer 2:
- Size: 32 neurons
- Components:
  - Weights (W): The first hidden layer has 64 neurons, and the second hidden layer has 32 neurons, and a weight matrix is used to establish this link. This avoids the situation whereby only some neurons in one layer affect those neurons in the next layer which is in the first hidden layer.
  - Bias (b): Biases are then added to weighted sum of inputs from the first hidden layer then the linear combination is again refined before feeding it to activation function.

- Activation Function: Another nonlinear activation function is then applied to the outputs of this layer enlarging the ability of the sine network to learn nonlinear relationship between the data.

➢ Hidden Layer 3:

- Size: 16 neurons
- Components:
  - Weights (W): The weight matrix is the one that links the second hidden layer and the third hidden layer. All the 32 neurons of the second hidden layer are linked to sixteen neurons of this third hidden layer.
  - Bias (b): The bias terms are incorporated to the outputs derived from second hidden layer; these aid in the fine tuning of the neuron's outputs before the onset of activation function.
  - Activation Function: Nonlinear activation function is applied again for making the mapping between the input and out complex.

➢ Output Layer:

- Size: 8 neurons
- Components:
  - Weights (W): The last weight matrix provides connection between the 16 neurons of the third latter hidden layer to the output layer. This enables the information which has been processed by the hidden layers to be reduced to 8 outputs.
  - Bias (b): A bias is added to allow the adjusting output after a put through activation function so that the outputs can be refined.
  - Activation Function: The output activation function varies depending on the sort of problem in the network. For classification problem, SoftMax function is applied to produce the probability between 0 and 1 of 8 different classes. As the name suggests, linear activation may be used for regression tasks.

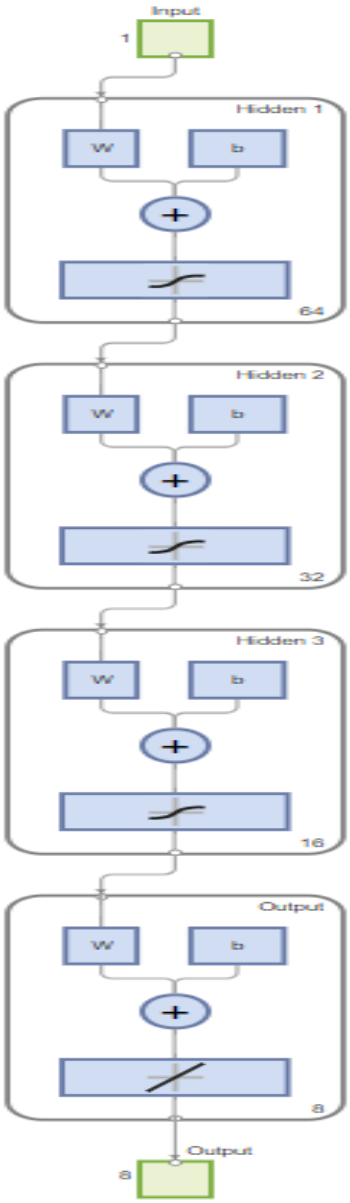


Figure 5.3: Feed Forward Neural Network for 3 hidden layers

Depth and Complexity:

The network also has moderate complexity since it has three layers of hidden nodes. This structure enables the network to discover more complex patterns of the input data as it spawns from one layer to the next. The first hidden layer has 128 neurons and with this practice the network should be able to represent as many patterns as possible from the input layer. The second hidden layer has neurons as 64 which lessen the feature space but is not significantly simple. Last of all, the third hidden layer which has only 32 neurons continues working with the data, highlighting the main characteristics of the input information before it will be transferred to the output layer. It offers the network the advantage of creating higher levels of abstraction maybe required in modelling more complex relations within a data set.

Layer Size:

The lesser neurons from 128 from first hidden layer to 32 in the third hidden layer is another usual tactic adopted to decrease the complexity of the map without losing important characteristics. This progressive reduction enables the network to identify relevant pattern whilst avoiding overfitting since all patterns are not significant. Hence by reducing the number of neurons across the layers the model can reduce the complexity of the input data and represent them in lower dimensions, which is very effective in the generalization of the model when tested on unseen data. This is done to ensure that the network captures complexities that may exist within the data but also make it simple to reduce risks such as over-fitting.

5.7.3. Feed Forward Neural Network for 4 hidden layers

Network Architecture

- Input Layer:
- Dimensionality: 1 neuron
- Description: In the input layer, there is one neuron to receive the scalar input feature. The network inputs only one feature at a time. It makes data entry easier and computational processing.
- Hidden Layer 1:
- Dimensionality: 128 neurons
- Components:
  - Weights W: This is the weight matrix connected to each of the 128 neurons in the first hidden layer from the input neuron. This matrix is important because it shows the relationship between the input data and a first set of internal computations.
  - Bias: Bias terms are added to each neuron in the hidden layer. This is very useful during better fitting of data, allowing the model to shift the linear combination of inputs and weights, relative to each neuron.
  - Activation Function: Non-linear activation is used on each neuron, typically in the form of ReLU or Sigmoid, allowing the network to learn complex and non-linear usage patterns within the data.
- Hidden Layer 2:
- Size: 64 Neurons
- Composition:

- Weights W: Weighted connections between all 128 neurons in the first hidden layer to the 64 in the second allow information captured in the last layer to be reduced into a more compact and feature-reduced format.
- Bias (b): A bias term is added to the output from the first hidden layer before passing through the activation function of the second layer.
- Activation Function: Also, the nonlinear activation function (ReLU or Tanh) allows complex data transformation to keep going on here.

➢ Hidden Layer 3:

- Size: 32 neurons
- Components:
  - Weights (W): The Activation weighting looks at how the 64 neurons transmitting from the second hidden layer connect to the 32 neurons of the third hidden layer. B This layer helps to enhance the learned features and at the same time, simplify them to handle them through subsequent layers.
  - Bias (b): Bias terms are incorporated to the output of the second hidden layer thereby giving more freedom in modelling the data. Activation Function: A non-linear activation function such as Rectified Linear Unit or Tanh function is applied on the neurons which makes them learn more non-linear relations.

➢ Hidden Layer 4:

- Size: 16 neurons
- Components:
  - Weights (W): There are 32 neurons from the third hidden layer, and it is connected to the 16 neurons of the fourth hidden layer. The third one is the final hidden layer, which restructures the features contained in the data in oneself format for the output layer.
  - Bias (b): Bias terms are incorporated including to adjust the weighted sum to better suit the transformation before the activation function is used.
  - Activation Function: To enable the network, capture the most complexities of patterns from the data, a nonlinear activation function such as ReLU is used.

➢ Output Layer:

- Size: 8 neurons
- Components:

- Weights (W): A weight matrix relates 16 neurons of the fourth hidden layer to 8 neurons of the output layer. These weights decide the mapping of the condense information of the concealed layers to the output layer.
- Bias (b): A bias term is incorporated to fine-tune the last weighted summation before the activating function is performed.
- Activation Function: In this layer, the activation function depends on the kind of problem that is being solved. It is appropriate for classification problems to apply SoftMax function, during which the network outputs the probability distribution of 8 classes. In regression tasks the output could be made continuous through using a linear activation function.

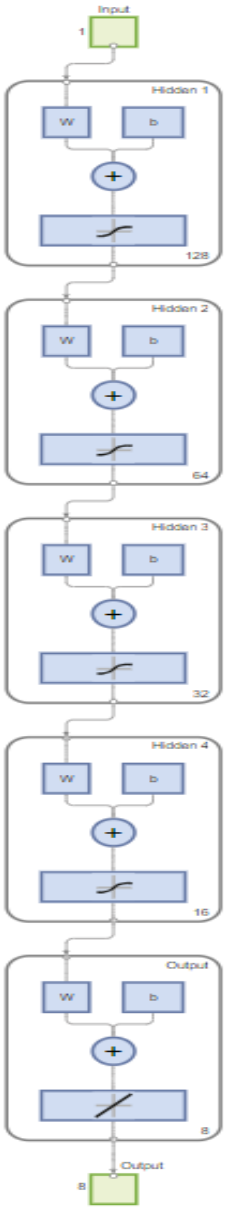


Figure 5.4: Feed Forward Neural Network for 4 hidden layers

Depth and Complexity:

The network in this case has four hidden layers hence a deeper network making it more capable of learning complex hierarchical patterns of the input data. For the first hidden layer with 128 neurons some first-level features of the input are desired. As the data progresses through the layers, the complexity of the features reduces: The second hidden layer with the 64 neurons identify the middle level of patterns, the third hidden layer with the 32 neurons identify relatively complex patterns and the last hidden layer with 16 neurons identifies the most basic and significant patterns before the data is passed to the output layer. This added dimensionality makes the network good for problems which involve the extraction of feature of various and complex levels such as image recognition or natural language processing.

Layer Size:

This reduction of neuron counts from one hidden layer to the other (128 → 64 → 32 → 16) was a deliberate effort to lower the dimensionality of the data progressively. The first layer starts with an extensive net that captures most of the patterns of the data and then progressively clarifies what is relevant and what is not towards the succeeding layers. This not only makes the model less complicated but also optimise the required features in the model by discarding the least significant or unnecessary information. This design enables the network to perform well on data not used in its training as it does no use all the complexity in the data as seen in the noise reduction and feature selection aspects of the design.

5.8. Neural Network Training

5.8.1. Neural Network Training for 2 hidden layers

Training Results and Progress

Training Results Training Completed:

- Min Gradient Reached: Training was cancelled because the gradient reached the minimum value specified in the training parameters. This suggests that further tuning of the artificial neural network weights will have negligible effect on reducing the error.

Training Progress

- Units:

- Epochs: The number of times the training algorithm has executed artificial neural network training using the entire data set.
- Elapsed Time: The amount of time required for training the designed artificial neural network
- Performance: Mean Squared Error (MSE).
- Gradient: The gradient of the performance function.
- Mu: The adaptive learning rate parameter for the Levenberg-Marquardt algorithm.
- Validation Check: The number of cases that failed to improve the validation performance.

➢ Initial:
- Performance: 1.59 (Initial MSE).
- gradient: 6.57.
- mu: 0.001.

➢ stopping value:
- epochs: 9.
- performance: 0.0449 (final MSE).
- gradient: 1.09e-15 (final gradient, below threshold).
- mu: 1e-9.
- validation check: 6.

➢ target values:
- epochs: 1.0e35.
- performance: 1e-6 (target MSE).
- gradient: 2e-15.
- Mu: 1e10.
- Validation Checks: 3e4 (max number of validations checks before stopping).

Training Algorithm

- Data Split: Randomly split the data into training, validation, and test sets.
- Training Algorithm: Levenberg-Marquardt Optimization algorithm (trainlm), efficient for medium sized networks.
- Performance: Mean Squared Error (MSE).
- Computation: Used MEX (MATLAB Executable) for efficient computation.

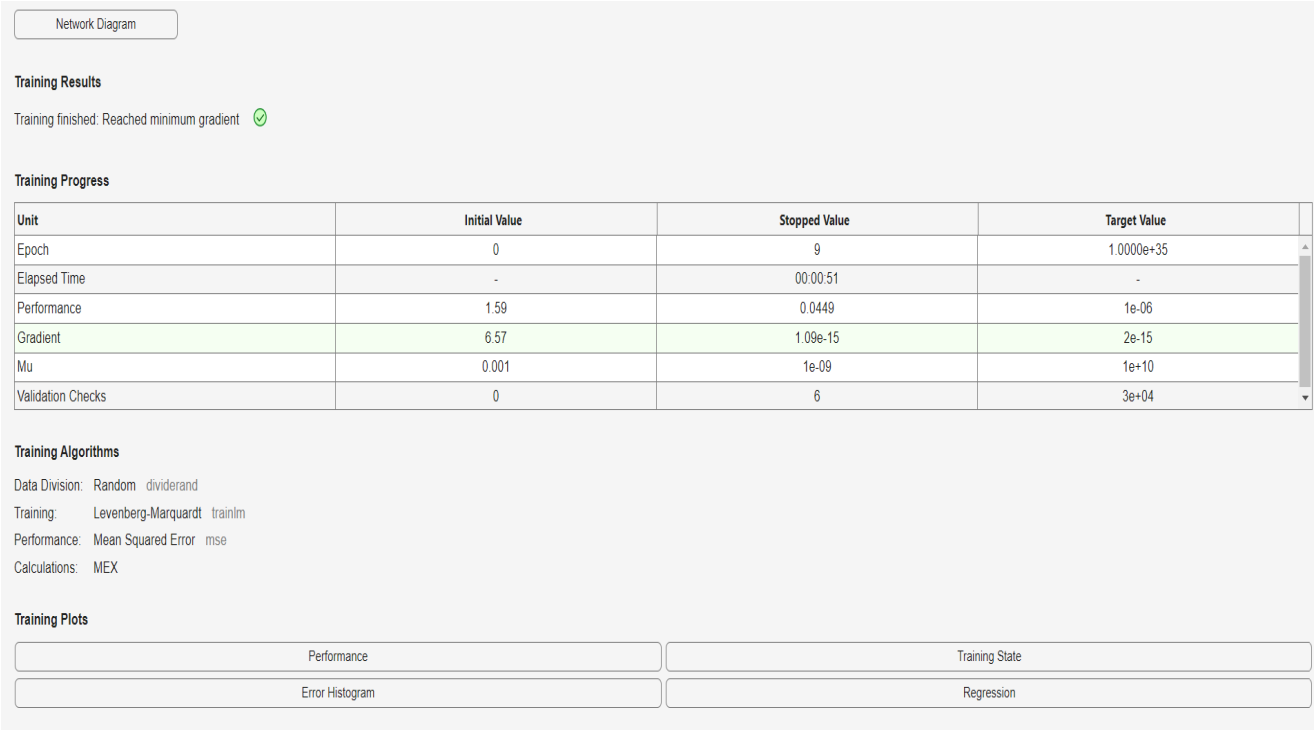


Figure 5.5: Neural Network Training for 2 hidden layers

Analysing the Results

- Performance Improvement: Performance improved from an initial Training MSE of 1.59 to 0.0449, indicating a significant error reduction with training.
- Epochs and Stopping Criteria: Training stopped after 9 epochs because the gradient reached a minimum threshold (1.09e-15) set at 2e-15. This suggests that the network has reached a point where further learning is minimal.
- Validation Check: The validation check reached 6, indicating that the model's performance on the validation data has not improved significantly after 6 epochs.

Inference: Training was efficient but stopped early because the gradient reached a minimum threshold instead of running for the maximum number of epochs. This indicates that the learning rate and other parameters were set appropriately, converging quickly to a solution. Generalization: The final MSE for training, validation and testing are close to each other, indicating that the model generalizes well to unseen data. Optimization algorithm: The use of the Levenberg-Marquardt algorithm (trainlm) was effective in minimizing error, as evidenced by the rapid decrease in MSE over the first few epochs.

5.8.2. Neural Network Training for 3 hidden layers

Training Results:

- Training Status: The training has been well done and the network attained the maximum value of mu which is an adaptive learning parameter. This mean that, increasing the training would not add great value in reducing the error rate.

Training Progress:

- Epochs: The training process was performed for 7 epochs, and it was mentioned that one epoch equals to a single pass through the whole dataset. By using input data, the network's weights and biases were updated seven times.
- Elapsed Time: The total training time was 18 seconds, of which showed that the training was very efficient.
- Performance: The evaluation criterion employed here is the Mean Squared Error (MSE), which quantifies the mean of the squared differences of the model's predicted and actual output. In the first time step the value of performance was 0. 357 and after training it reduced to 0. 0371. The target performance value on the other hand was established at 1. 0e-06, this indicated that while the network capability had been improved, it was just a shade off from what is a very small error margin.
- Gradient: It signifies the first order derivative of the performance function, namely the mean square error, with respect to the weights. It started at 1. 6, training stopped when the gradient became very small and reached the value of 3. It remains as 12e-13 that meant the model requires minimal tweaks to be made to gain more accuracy in its predictions.
- Mu (Levenberg-Marquardt parameter): Beginning at 0, Mu continued to an existing value. Number of respondents at the beginning of the study with code 001 rose to 1 as seen in table 3 below. The people will reach 0e10 by the end of training. In the Levenberg-Marquardt algorithm, mu is used to control learning rate. The results also showed that a high value of Mu meant that the model had a problem of concentrating on a higher value that was closer to the global optimum, and hence, the training stopped.
- Validation Checks: Just to remind you the validation checks have gotten through an optimistic count of 4. This metric implies that training ceased because there was no improvement of the validation loss for the past four checks to avoid overfitting of the model to the training data.

Initial Values:

- Performance MSE starting was 0.357, which is the level of error prior to the network having started optimizing.
- Gradient: The gradient started at 1.6, showing how much the initial predictions were from target values.
- Mu: The Mu starting was 0.001, which is the standard value for starting any type of Levenberg-Marquardt optimization.

Stopping Values:

- Epochs: The training stopped after 7 epochs.
- Performance MSE: The final MSE of 0.0371 was due to great reduction starting with the initial error.
- Gradient: The final gradient, 3.12e-13, is quite close to the target threshold of 2e-15 and promises that further tweaks will have little effect on the general accuracy of the model.
- Mu: Mu has reached the maximum value of 1.0e10; that means the network had reached such a point where further learning was inefficient.

Target values:

- Performance-MSE: Set target MSE at a very low value of 1e-06. The model run improved substantially but did not quite reach this target. Gradient Target set at 2e-15, showing a very tight tolerance for the rate of improvement. Mu Maximum allowable Mu set at 1e10, which is the value the network reached at the end of training.
- Validation Checks: The model was allowed a maximum of 3e04 validation checks before stopping, but training ended after only 4 checks.

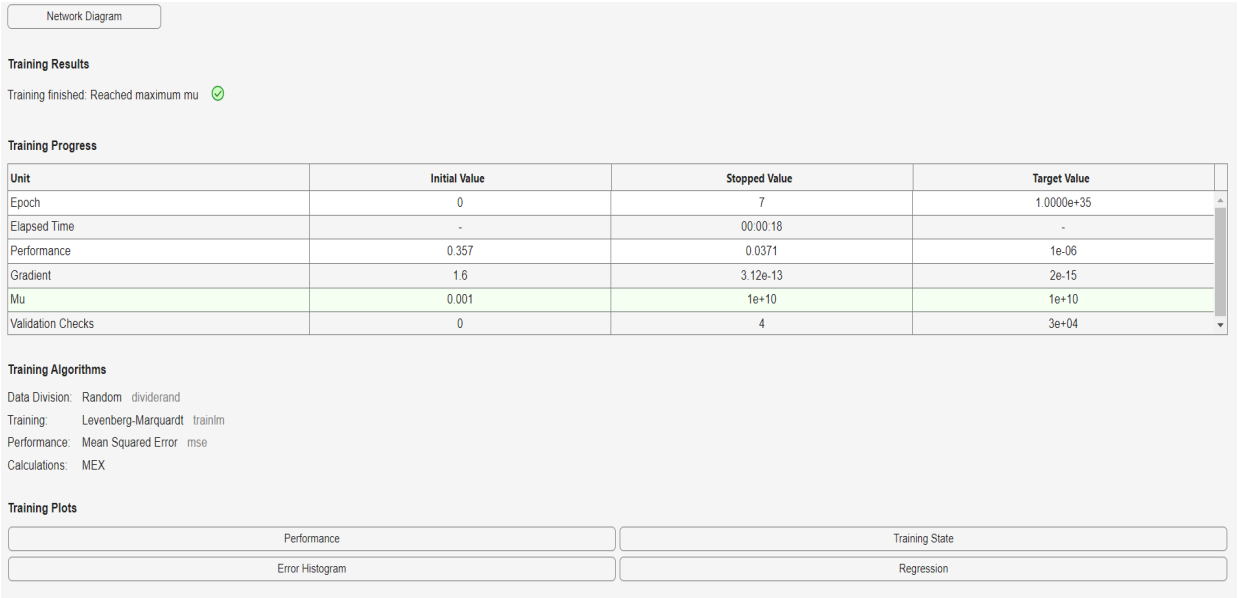


Figure 5.6: Neural Network Training for 3 hidden layers

Training Algorithm:

- Data Division: The data were divided randomly into training, validation, and testing. This is quite a standard method of ensuring the model gets trained on one portion of the data, validated on another against overfitting, and finally tested on the remaining part to appraise performance.
- Training Algorithm: The optimization algorithm applied was the Levenberg-Marquardt (trainlm). This is good to go for medium-sized networks and balances training speed with stability, when the network is converging to a final solution.
- Performance Metric: The performance metric of the model was that of Mean Squared Error-MSE with an optimality of minimum average of squared differences between predictions and actual values.
- Calculations: MEX-the method used by the network to accelerate calculations within MATLAB-was done to increase computational efficiency in MATLAB.

Analysis:

- Performance Improvement: The network has exhibited a notable increase in its performance, due to the training performed, and is capable of learning correctly as shown by the strong convergence, disclosed by a decreased error from MSE 0.357 to MSE 0.0371 (where lower is better). The error has been reduced quite a lot, which was a good thing. Unfortunately, the target error of 1e-06 was still not reached.
- Stop Criteria: The gradient was too low, so gradients were near the local minimum after 7 epochs of training, meaning minimal improvements afterward. Moreover, the Mu parameter was in the order of 1e10, which represents an increase in the convergence of the training. You would see changes in the training loss as you optimize the training by adding more data to the model. But, as validation checks terminated at 4, you would not experience any further progress with them. In other words, this is a good stopping place because we have reached a plateau on the validation data set, and attempting to continue would likely lead to overfitting.

5.8.3. Neural Network Training for 4 hidden layers

Training Results:

- Training Status: Network converged to the minimum gradient threshold; hence, network is TRAINED. This supports the idea that, for all practical purposes, this training has already converged, and further optimization, if continued, would yield diminishing returns.
- Epochs: Training stopped after 9 epochs-meaning the network went through 9 cycles of updating weights and biases to reduce error based on the data.
- Elapsed Time: The total training time took 4 minutes and 17 seconds hence, the moderate complexity of the data or model size. Performance-MSE: Initial MSE was 0.199, after training it is 0.0231; though it gives good improvement, it hasn't reached to the target value 1e-06.
- Gradient: The gradient, or the rate of change of the performance function with respect to the weights, starts at 0.86 and goes as low as 5.24e-16, meaning no more meaningful changes can be further done.
- Mu (Levenberg-Marquardt parameter): Mu, starting at 0.001, only grew to 1e-08, and that gives evidence of near convergence.
- Validation Loss: Validation checks came in 6 times, but training stopped when the validation error stopped decreasing to prevent overfitting. Initial Performance Results-MSE: Initial MSE was 0.199-a reasonable value.
- Stopping Values: At the end of 9 epochs, training was stopped, the ending MSE improved to 0.0231, having greatly improved but still rather far from the target of 1e-06. The ending gradient was 5.24e-16, so minimal more improvements were possible. Mu finally reached a value of 1e-08, showing that the network has fine-tuned its learning rate.
- Target Values: The target performances were 1e-06 and were not reached; the target gradient was 2e-15. Due to these factors, model training stopped after 6 validation checks to prevent overfitting.

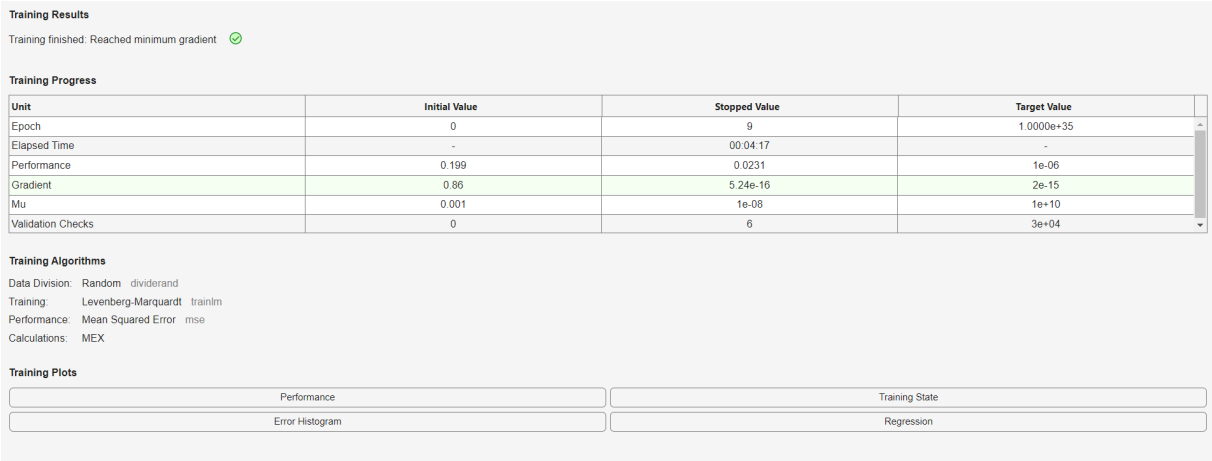


Figure 5.7: Neural Network Training for 4 hidden layers

Training Algorithm:

- Data Division: The data are divided randomly into training, validation, and testing sets so that the network generalizes well and does not fall into overfitting problems. Training Algorithm: Employ the Levenberg-Marquardt backpropagation (trainlm), which is the right choice for training medium-sized networks and is known for speed and robustness.
- Performance Metric: Performance is measured in terms of mean square error (MSE), the standard performance metric given by the average of squared differences between predicted and true values.
- Computation: To speed up the computation, MEX was used and the training in MATLAB was accelerated.

Discussion of Results

- Progressive Performance Improvement: The MSE has dropped considerably from 0.199 to 0.0231, which is a great drop. However, it has not achieved the target performance of 1e-06, which means that further training is required, or the complexity of the data is beyond the capability of the network.
- Stopping criteria: The training was stopped after 9 epochs since the gradient had dropped to a very small value of 5.24e-16, which hinted toward almost complete convergence of the model. Also, a low Mu value of 1e-08 depicts that the network has come to that point it had little to improve on.
- Validations Checks: The number of validation checks are up to 6 which implies that validation error has plateaued for 6 epochs in total, ensuring no overfitting of the model.

5.9. Training Record

5.9.1. Training Record for 2 hidden layers

This plot in figure 5.10 shows the performance of the neural network (mean squared error (MSE)) over nine epochs for the training, validation, and test sets. The y-axis has a logarithmic scale, showing a wide range of MSE values.

Key Observations

- Initial Performance (Epoch 0): All three sets (training, validation, test) start off with a high MSE of approximately 10^0=1 to 1.2. This is expected since the network starts with random initial weights.
- Rapid Decrease (Epochs 1-3): We see a rapid decrease in MSE for all three sets over the first few epochs.
- Plot (Epochs 4-9): After approximately epoch 3, performance begins to be stable. The MSE values for the training, validation, and test sets stabilize at approximately 10^{-1} to 10^{-2}. This suggests that the network is not making much progress in reducing the error.
- Final Performance: The final performance values are nearly the same for all three sets, indicating good generalization. There is no significant gap between the MSE for training, validation, and testing, which is a positive sign that the network is not overfitting.

Inference: The close MSE values for the training, validation, and testing sets indicate that the model generalizes well to unknown data. This means that the neural network has learned how to recognize received multiplexed message signals, rather than just memorizing the training data. Early Stopping: The model seems to reach a stable performance after a few epochs. This plot suggests that including additional training epochs may not yield significant performance gains, consistent with an early stopping strategy. Given the fast convergence, the chosen learning rate seems appropriate for this problem. Model performance: The final MSE is about 0.0449, indicating that the model detects the transmitted signal accurately despite the noise.

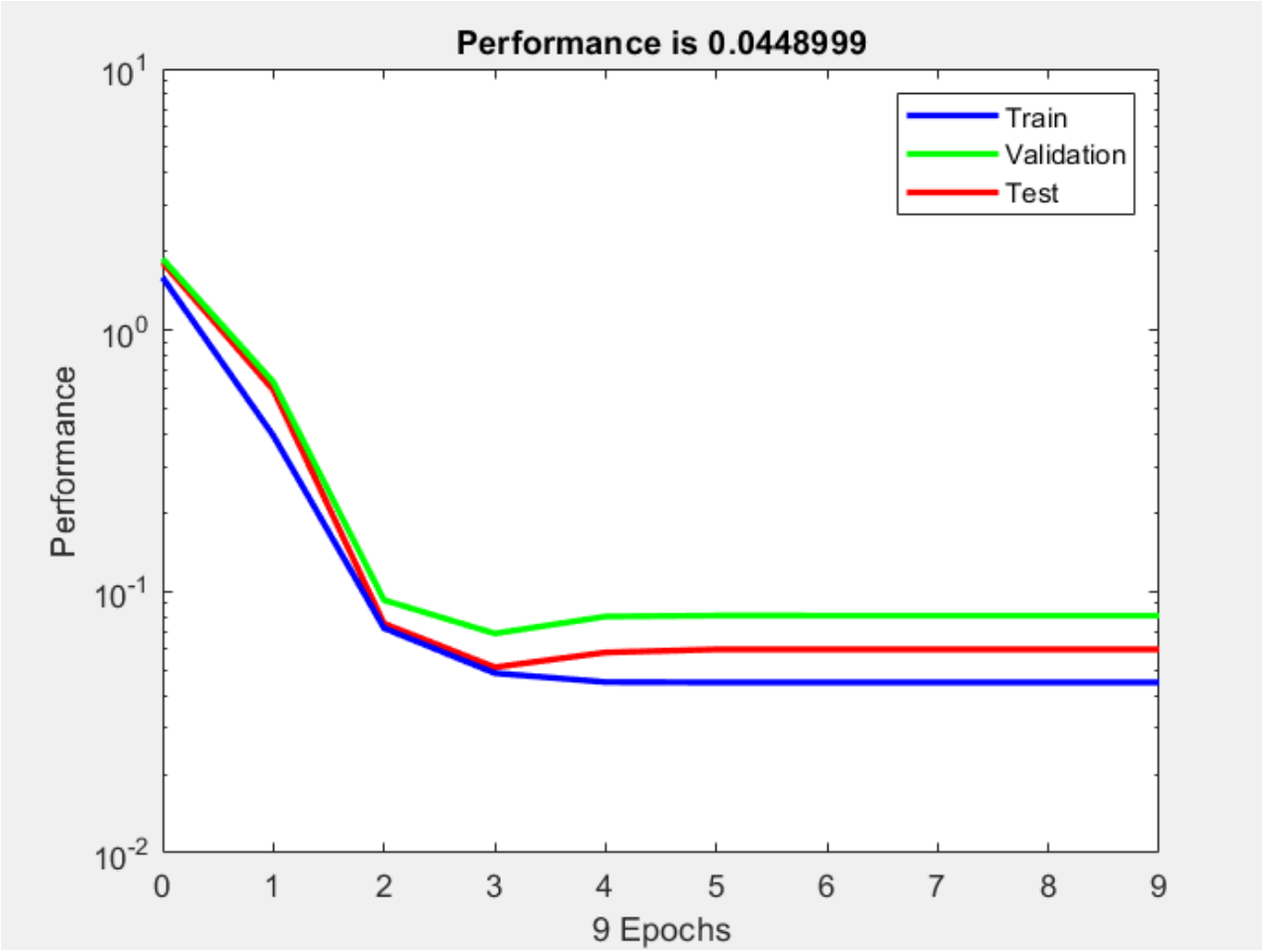


Figure 5.8: Training Record: Performance VS Epochs for 2 hidden layers

The final MSE is about 0.0449 being very close for the training, validation, and test sets. This shows that the network is quite successful in recognizing the transmitted signals among noise. The very close values of MSE training, validation, and test sets suggest that it is a strong indicator of good generalization which is to say that the network has not overfitted to the training data and thus can efficiently deal with new and unseen data

1. Training, Validation, and Testing MSE:

- Training MSE: It could be seen that the network is showing what it has learned from the training data through the decrease in training MSE. A decline in the training MSE is indicative of the network's capability to learn patterns and establish relationships that may be present in the training dataset.
- Validation MSE: The loss is an indicator of the performance of the network on the validation set and hence can be used to evaluate if the model is able to generalize to new data. The validation MSE is stable, so the network is not overfitted and does generalize the patterns it learns well.

- Testing MSE: The network can use the test set of data to predict what would be the actual performance of the network on the future test datasets so that the generalizing capability of the network is ascertained. The fact that MSE values for the test sets are similar suggests that the model will perform consistently and be reliable.

2. Realization of Multi-User Detector (MUD):

It is apparent from the low final MSE that the neural network possesses the ability to perfectly separate and decode signals of multiple users, even under the adverse condition of noise. The accuracy of this operation known as MUD is essential because the ability of the network to separate users and correctly interpret the signals sent by overlapping users has a direct impact on the performance of the network.

3. Enhancing Spectrum Utilization:

The detection and precise reconstruction of signals power the effective utilization of the spectrum resources. The network can, therefore, differentiate among users' signals and properly allocate the spectrum, minimizing the interference and maximizing the usage of the available bandwidth. The improved capability is further used by the network to support multiple users and applications at the same time, which eventually leads to the enhancement of network performance and the satisfaction of the users.

Explanation of Training Plot for Mean Squared Error (MSE)

This plot in figure 5.11 shows the Mean Squared Error in the training, validation, and testing sets as the neural network goes through its training epochs (cycles of learning).

- X-axis (Training Epoch): These are the number of epochs the neural network goes through to update its weights.
- Y-axis: Mean Squared Error - MSE means the average of the squared differences between predicted and actual values. The lower the value of MSE, the better the performance.

➢ Observation Points:

- Initial Error Epoch 1: MSE for the training set was at 1.59143, 1.81558 for the validation set, and 1.87448 for the test set initially. These are high values, indicating this model had a large error in its initial sets of predictions, as usually happens before the training optimizes the weights.

- Sharp decline: There is a steep decline in the MSE across all sets between epochs 1 and 3. By epoch 3, the MSE falls below 0.2 for both training and validation. This means that the network had lots of quick improvements, by which it rapidly learned from the data.

➢ Plateau Phase - Epochs 4-9:

After epoch 3, all the values for training, validation, and testing start to flatten out at an overall low value. At epoch 10, the training MSE comes in at 0.044899, while validation goes as low as 0.0600062, with testing settling at the same low rate; this will provide sufficient indication that the network has converged, and further training does not result in significant error reduction.

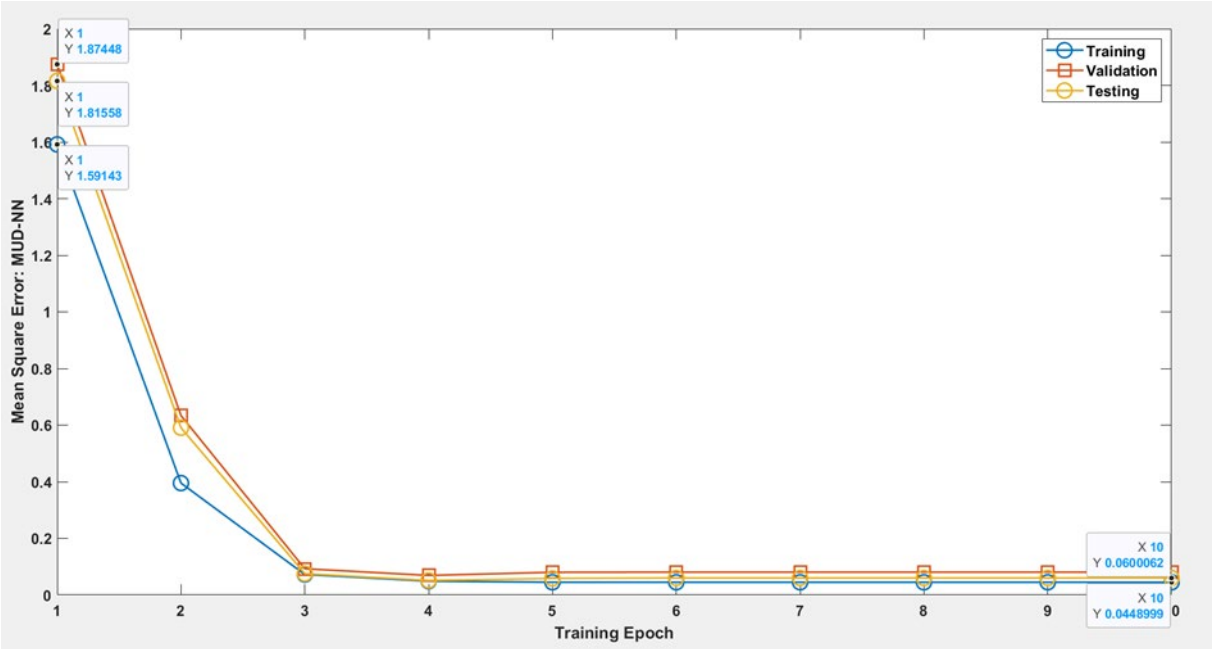


Figure 5.9: Training Record: MSE MUD-NN VS Epochs for 2 hidden layers

Results Analysis:

- Convergence in less time: This plot will depict the convergence of the network, which is rapid. Most of the gains are found during the initial few epochs, after which there is not much gain because the training process brought down the error effectively at the beginning.
- Minimal overfitting: The training, validation, and testing errors remain close to each other throughout the process, with no significant divergence. That is a very good generalization from the model for unseen data, and this does not point toward overfitting, when a model performs well on the training data but performs terribly on new data.

- Effective Early Stopping: Training stopped after 9 epochs because the gradient reached its minimum. Indeed, looking at the stability of the validation error from epoch 4 would assume that further training beyond this point would probably be useless.

5.9.2. Training Record for 3 hidden layers

The following plot in figure 5.12 summarizes the performance of the neural network, in terms of Mean Squared Error (MSE), as a function of epoch for the training, validation, and test sets. This plot uses a logarithmic y-axis scale to effectively show a range of MSE values spanning several orders of magnitude.

Key Observations

- Performance at Initial Solution (Epoch 0): The training, validation, and test MSE all start high at about the magnitude of 10^0 or 1. In general, such a high starting error is expected because the network is initialized with randomly set weights.
- Sharp Drop (Epochs 1-2): Over the first two epochs, there is a very rapid drop in the MSE recorded over the three datasets. This is indicative that the network is learning well off the data and correcting its weights such that the error drastically comes down.
- Stabilization-Epochs 3-7: After the sharp drop, the MSE values from all the datasets gradually stabilize. At about epoch 3, all the values have reached an approximate level of 10^{-2}, after which only minor variations can be observed in further epochs. Thus, the network has optimized its predictions on the grounds of the training data to such an extent that further additional training does not significantly improve the performance.
- Final Performance: The final MSE values of the training, validation, and test datasets are all very close and around 10^{-2}. This itself is a strong indication of good model generalization because it means that the network is doing similarly well on the seen data-training-and the unseen data-validation and testing-which in essence means it is not overfitting to the training data.

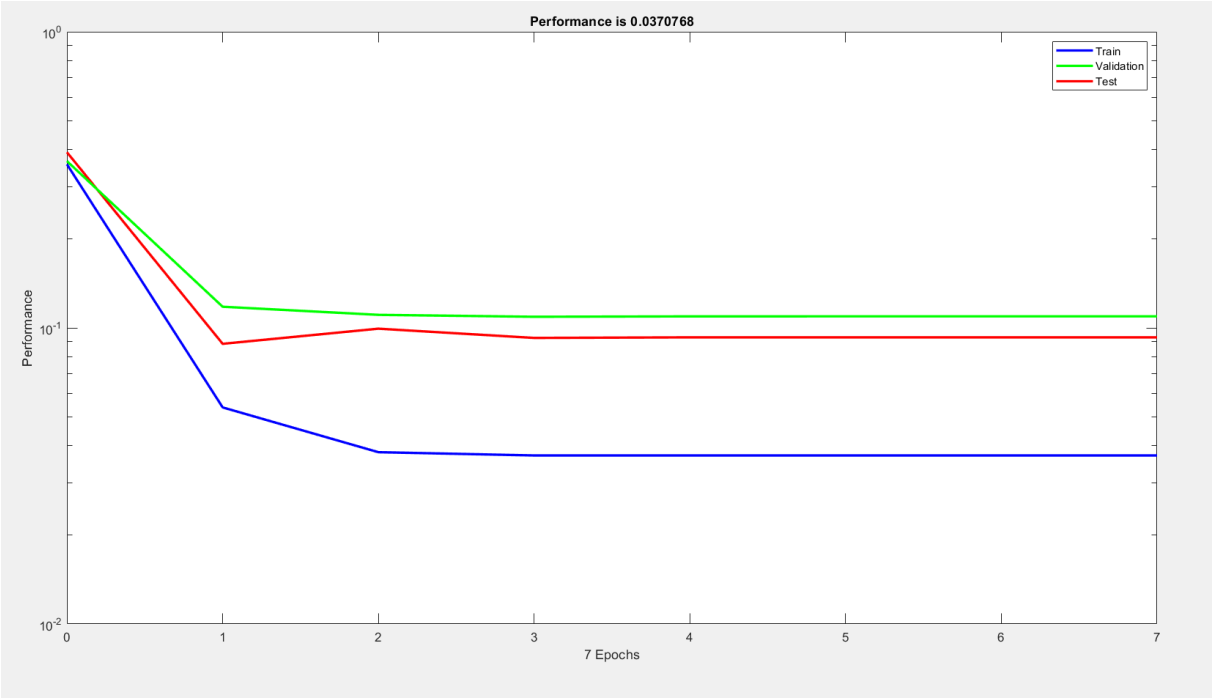


Figure 5.10: Training Record: Performance VS Epochs for 3 hidden layers

This graph in figure 5.12 shows that the neural network has undergone significant learning within a few epochs, as evidenced by the rapid improvement in performance, followed by the stabilization of this process, whereby subsequent epochs yield little error reduction. The convergence of MSE values across the training, validation, and testing datasets by the end of this training shows good generalization across different datasets. This is the optimal behaviour for neural network performance, the perfect tuning in the learning process that balances efficiency in learning and capability to generalize to new, unseen data. The fact that divergence between training and other datasets does not happen toward the final epochs reassures that the network does not memorize the training data but instead learns to predict new instances effectively.

Explanation of Training Plot for Mean Squared Error (MSE)

This plot in figure 5.13 shows that the neural network has good learning efficiency and, hence, good generalization capability for unseen data, supported by the rapid stabilization of this curve and the almost consistent MSE values between the sets. It is proper to say that the settings of the learning algorithm and model seem quite adequate to solve this task.

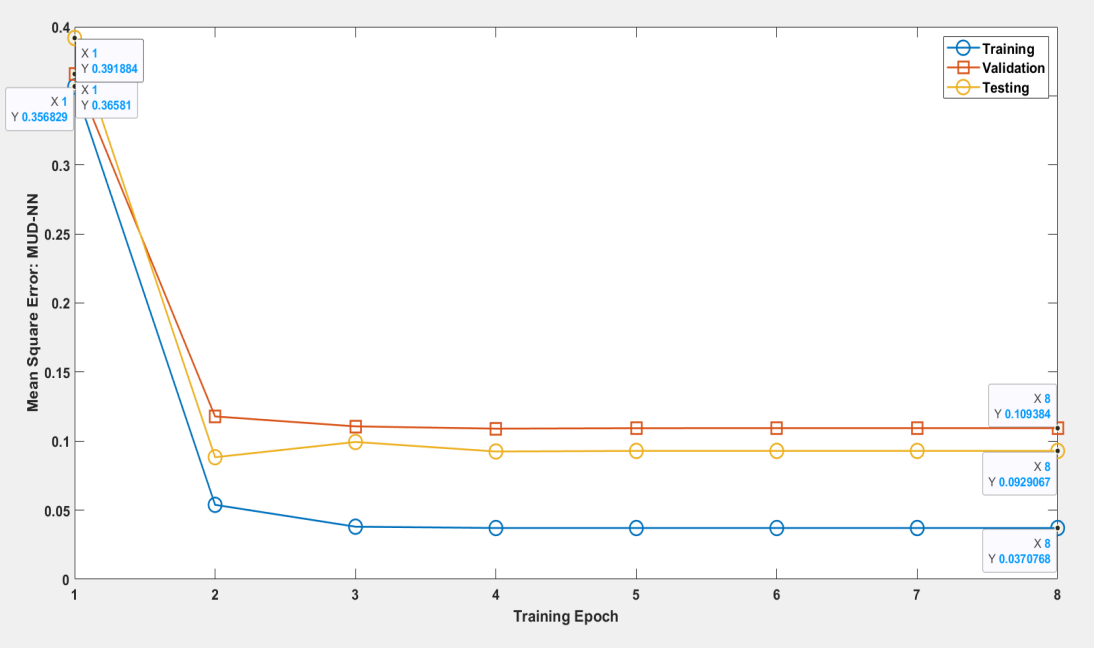


Figure 5.11: Training Record: MSE MUD-NN VS Epochs for 3 hidden layers

The new chart now represents the Mean Squared Error's trend of training, validation, and test datasets while training a neural network for 8 epochs. Some of these key elements and observations are further elaborated upon in the description.

- Plot Breakdown
- X-axis (Training Epoch): This reflects the number of training cycles run on the model- a total of 8 epochs in this plot.
- Y-axis (Mean Squared Error - MSE): It depicts the values of MSE, which means the lesser the value of MSE, the better the predictive accuracy by the model.
- Observation Points
- Initial ErrorEpoch 1: The MSE starts with high values of training at about 0.392, validation at about 0.356, and testing at about 0.357. These starting errors characterize how poor the model was initially with the prediction, insofar as at this stage the model parameters have not yet been optimized by learning.
- Sharp Decline: Epochs 1-2: This sharp decline in MSE across all the datasets from epoch 1 to epoch 2 means that within this very short time of training, the model quickly learned from its initially trained data and that its performance has strongly improved.
- Plateau Phase (Epochs 2 – 8): After the steep decline, the MSE of all the data sets becomes constant and remains very low from epoch 2 onwards. More specifically, it would stabilize to about 0.071 for training MSE, about 0.109 for that of validation, and about 0.091 for testing toward the end of epoch 8. This will lead to stability in the sense

that the model has greatly optimized its performance, and further training results in little improvement.

- Discussion of Results

- Fast Convergence: This rapid improvement in MSE in the initial epochs represents great efficiency of the neural network in learning the data. Most of the learning happens in the initial epochs.
- Generalisation Capability: Closely tracking the MSE values of training, validation, and testing datasets during the training epochs insinuates good generalization. The model does well consistently on different sets of data, which is indicative of its robustness against overfitting.
- Justification of Early Stopping: The MSE values have stabilized soon and have been quite flat across the datasets, which suggests that early stopping might be effective. It will prevent overfitting and avoid useless computational costs since further epochs add barely any value in performance.

5.9.3. Training Record for 4 hidden layers

This plot in figure 5.14 represents the performance of a neural network on the training, validation, and test datasets over nine epochs. Note that the y-axis is on the log scale. This lets us easily see the decrease in MSE over several orders of magnitude. Below are explanations for some key features to look at in this plot:

- Initial Performance (Epoch 0): First, all three sets have an MSE value in the order of magnitude of 10^0 to 10^1. That is expected since at random starting weights, the network hasn't picked up anything from the data yet.
- Sharp Drop-off: There is a sharp decline in MSE for all sets in the initial few epochs. This phase reflects that the network is learning fast from the training data and greatly reduces the error while adjustments of the weights and bias are being made based on the backpropagation of errors.
- Performance Stabilization (Epochs 4-9): After such an initial steep learning phase, MSE converges to a constant value for all the datasets, oscillating between 10^{-1} and 10^{-2}. That plateau might indicate that most of the low-hanging fruit from the initial learning have been captured and that further error reduction is incrementally getting tougher.

- Final Performance: By epoch 9, which is the last epoch in this MSE training, validation, and test dataset values converge within a very similar level in the range of 10^{-1} to 10^{-2}, showing good generalization. These curves being closely upper bounded at the end and not showing much divergence toward the end signifies that the model has been tuned well regarding what it has seen and neither overfits nor underfits.

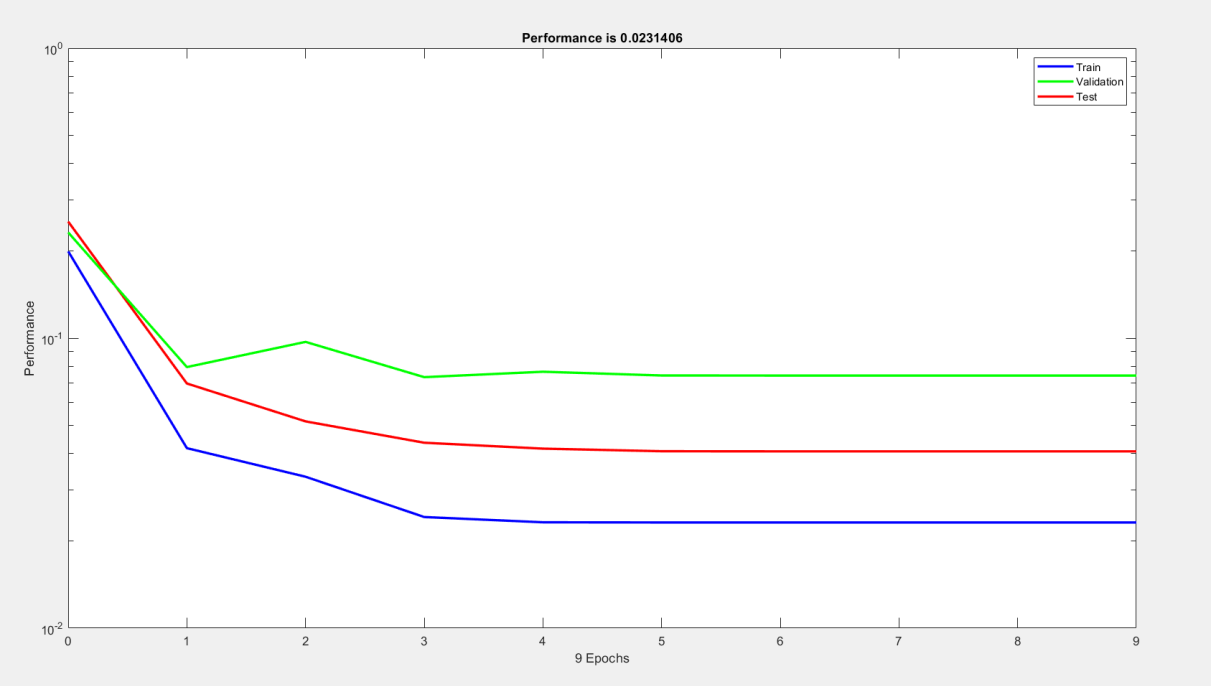


Figure 5.12: Training Record: Performance VS Epochs for 4 hidden layers

In other words, summarizing, what this graph is telling us is that it's a neural network which learns very well and generalizes very well into the unseen data. The performance is constant across all data for all epochs of training after the first few epochs.

Explanation of Training Plot for Mean Squared Error (MSE)

This plot in figure 5.15 shows the performance of the neural network mean squared error (MSE)) over 10 epochs for the training, validation, and test sets. The y-axis has a logarithmic scale, showing a wide range of MSE values.

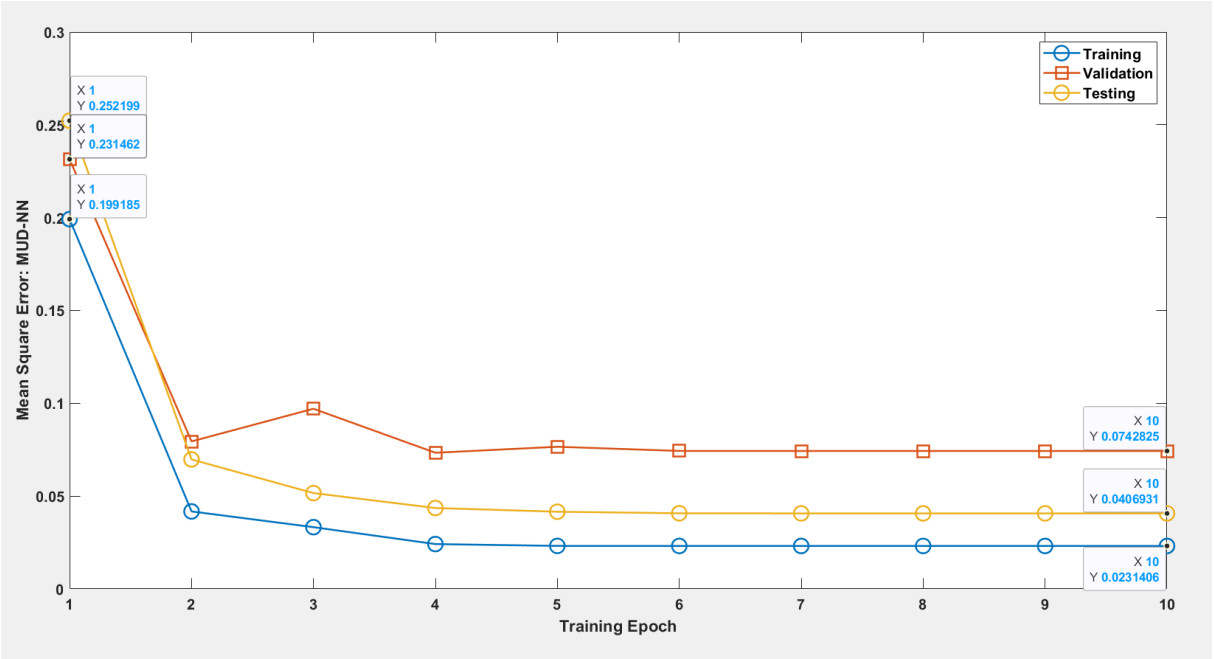


Figure 5.13: Training Record: MSE MUD-NN VS Epochs for 4 hidden layers

➢ Graph Explanation

- X-axis (Training Epoch): Charts the number of times the model has gone through training on the data; a total of 10 epochs are used.
- Y-axis (Mean Squared Error - MSE): Plotted as the MSE, which basically is the average of the squares of the errors in prediction. The smaller this number is, the better the model performs.

➢ Observation Points

- Initial Error (Epoch 1): First, MSE across all the datasets starts high: training set at about 0.252, followed by the validation set about 0.231 and testing at about 0.200. That would seem the predictions at this stage are not quite accurate, which should be expected before the model has adjusted from learning of the data.
- Sharp Decline: It may be noticed that there is a big drop in MSE across all the datasets between epoch 1 and 2. After the third epoch, MSE values plummet and start getting stabilized. That indicates that the learning has happened rapidly, and the performance has improved substantially in the initial phases itself.
- Plateau Phase (Epochs 3-10): After the sharp drop, all MSEs settle down and remain roughly constant. By epoch 10, the training MSE is at about 0.074, the validation about 0.046, and the test close to 0.023. This plateau indicates that the model has roughly converged with very slight further improvements.

➢ Analysis of Results

- Fast Convergence: The graph below illustrates fast learning within the network, and this can be inferred from the fact that large drops in MSE happen within fewer epochs. The steeper the rise, the stronger was the improvement, which means the model has been trained well.
- Minimal Overfitting: Close tracking of the MSE values across training, validation, and testing shows that the model generalizes well across different sets of data. Considering how small the difference between these curves is, overfitting for the model is not there; that is, in essence, it does not just memorize the training data but learns to predict unseen data effectively.
- Early Stopping Works Effectively: Starting with the first few epochs, model performance stabilizes, and beyond roughly epoch 3, there is hardly any notable improvement in MSE. This stability can provide an indication that increasing the training beyond 10 epochs might not help much; hence, a decision for early stop might be done to prevent overfitting and unnecessary computation.

That would indicate that the learning process is well tuned: the neural network picks up the training data fast and generalizes well to previously unseen data. Early and rapid improvement with subsequent stability in performance means that this is a valid regime of training with a good choice of model architecture for the task at hand.

CHAPTER SIX: CONCLUSION AND RECOMMENDATION

This research proposes a machine learning algorithm whose use has the capability of enhancing performance of wireless networks while reducing costs. This research showed that approaches based on machine learning yield promising results and that this field has further research pathways. Possible improvements from this research include such aspects of research which could be used in other fields of research in the future, and they can also revolutionize the future design and use of wireless networks.

In this study, we identified the method and means of improving the spectrum usage in the hybrid NOMA network with the use of the deep neural network (DNN). The work was motivated with a view of enhancing performance in terms of spectral efficiency as well as user detection in challenging multiuser networks. Our approach involved several key components: pioneering spreading codes, modulating user's message, and utilizing the machine learning approach to improve the Mult-user detection technique.

First, the spreading codes and the random messages were generated to create an environment of spread communication system. The specific modulation techniques that were used comprised of 8PSK and other instances of PSK as well as QAM to suit different users. These modulated signals were transmitted through a Multiple-Input Multiple-Output (MIMO) channel to emulate the actual signal transmission environment. For the training phase, feature extraction and data dimensionality reduction were done by employing the Singular Value Decomposition (SVD) for determination of the main components of the data.

The Deep Neural Network base referred to as DNN architecture was well developed, with special attention given to Channel while building the base, the design goal being to have an artificial neural network that can execute multi-user detection. Based on the mentioned inputs, we proposed a feedforward neural network with a predetermined numbers of layers and neurons, including the use of activation functions to aid learning process. Training was done by passing the modulated messages through the network in which data was segmented and normalized for learning. This was achieved by testing performance metrics using the Mean Squared Error (MSE) of training, validation and the test phases while

giving an indication of the ability of the model in predicting and enhancing spectrum utilization.

Therefore, this research proves that the deployment of DNNs into NOMA networks can considerably improve the spectrum efficiency as well as the user's detection. It was found that using the SVD technique for feature extraction and dimensionality reduction provided a useful input data for training, and DNN approach might be helpful in dealing with multiple users. From the results obtained in this study, future work should consider using more complex architectures of neural networks and different optimization methods. Moreover, expanding the work to use real life situations rather than simulating and analysing the effect of different channel conditions may offer a greater perspective of how the proposed technique may be applied. All in all, these findings open the way for developing improved architectures and more efficient algorithms in modern systems of wireless communication.

REFERENCES

1. Ahn, J. (2019, FEBRUARY). Sparsity-Aware Multi-User Detection for Massive Machine-Type Communications.
2. Bal, A., Nuhi, J. F., & Mousumi, S. (2021). A Comparative performance analysis among Hybrid NOMA Schemes for 5th Generation (5G) and beyond wireless communications.
3. Barbé, M. M. (2020). Development of MATLAB application to generate NOMA-CAP 5G signals.
4. Chaowei Wang, Mingliang Pang, Gaofeng Cu, Xinshi Chang, Fan Jiang, Yuan Yao, & Weidong Wang. (2023). Joint Waveform Design and Multi-user Detection in Symbiotic Ambient Backscatter NOMA Systems.
5. Chuan, Z., Yuan-Hao, H., Farhana, S., & Zhongfeng, W. (2017, DECEMBER). Advanced Baseband Processing Algorithms, Circuits, and Implementations for 5G Communication. 7, 14.
6. Fang, F. (2017). Energy Efficient Resource Allocation for Non-Orthogonal Multiple Access (NOMA) Systems.
7. Fatima, H., Rasheed, H., Syed Ali Hassan, H., & Ekram, H. (2020, QUARTER). Machine Learning in IoT Security: Current Solutions and Future Challenges.
8. GUIYONG, Z. (2017). Compressed Sensing Based Block Type Multi-User Detection for Sporadic IoT Communications.
9. JEON, N. (2021, May). Active user detection and low complexity multiuser detection for Unsourced multiple access.
10. Jingjing Cui, Zhiguo Ding, Pingzhi Fan, & Naofal Al-Dhahir. (2018, NOVEMBER). Unsupervised Machine Learning-Based User Clustering in Millimeter-Wave-NOMA Systems.
11. Kader, M. F. (2019). A power-domain NOMA based overlay spectrum sharing scheme.
12. Kaiyuan, J., Jiawei, Z., Haibin, W., Aili, W., & Yuji, I. (2020, February 9). A Novel Digital Modulation Recognition Algorithm Based on Deep Convolutional Neural Network. 15.
13. Lenovo, P. (2024, August 28). ThinkPad Laptops and Mobile Workstations.
14. Myo, M. M. (2020). Cooperative Relaying System employing OFDM based NOMA system.
15. OHTSUKI, T. (2022, August 10). Machine Learning in 6G Wireless Communications.

16. Omer Mohammed, K.-D., Aymen Mohammed, K.-D., Maiduc, O. A., & Mohammed Khodayer, H.-D. (2023). Strategy for Non-Orthogonal Multiple Access and Performance in 5G and 6G Networks.
17. Peisen Wang, Neng Ye, Jianguo Li, Boya Di, & Aihua Wang. (2022, OCTOBER). Asynchronous Multi-User Detection for Code-Domain NOMA: Expectation Propagation over 3D Factor-Graph.
18. REBHI, M. (2022). Contributions to non-orthogonal multiple access techniques for massive communications.
19. Thi, T.-X. C. (2023). Singular value decomposition and applications in data processing and artificial intelligence. HPU2 Journal of Sciences, 02(03), 9.
20. Vineela, K., & Madhu, R. (2018, January). Performance Analysis of various Modulation Schemes on Inter Satellite Communication Link. International Journal of Computational Engineering Research (IJCER), 08(1), 13.
21. Vivienne Sze, Yu-Hsin Chen, Joel Emer, Amr Suleiman, & Zhengdong Zhang. (2017, Oct 17). Hardware for Machine Learning: Challenges and Opportunities. 9.
22. Wei, Z. (2019, June). Performance Analysis and Design of Non-orthogonal Multiple Access for Wireless Communications.
23. Wojciech Samek, Slawomir Stanczak, & Thomas Wiegand. (2017, October 13). The convergence of machine learning and communications.
24. Xiaojie Wang, Chulong Liang, Li Ping, & Stephan ten Brink. (2020, MAY). Achievable Rate Region for Iterative Multi-User Detection via Low-Cost Gaussian Approximation.
25. Xiaojuan, Z., Shouyi, Y., Aihua, Z., & Xiaoyu, L. (2018). A Compressive Sensing Based Multi-user Detection Algorithm for SIMO-NOMA Systems.
26. Xuan, L., Hongmin, L., Yifeng, H., Fulin, W., Chengxi, Z., & Xiaoliang, W. (2023, November 12). Analysis on the Effect of Phase Noise on the Performance of Satellite Communication and Measurement System. 17.
27. Yaohua Sun, Mugen Peng, Yangcheng Zhou, Yuzhe Huang, & Shiwen Mao. (2019, FOURTH QUARTER). Application of Machine Learning in Wireless Networks: Key Techniques and Open Issues.
28. Yue, L., Ning, C., Minghe, M., & Gang, L. (2020, September 28). Shaped offset 8PSK and coded shaped offset 8PSK with improved spectrum efficiency for satellite communication. 12. doi: 10.1049/cmu2.12062
29. Zhao, X., Yang, S., Zhang, A., & Li, X. (2018). A Compressive Sensing Based Multi-user Detection Algorithm for SIMO-NOMA Systems.

APPENDICES

APPENDIX A: Generated Random Messages

```
2.Generated Random Messages:
      1      1      0      0      0      1
      1      0      1      1      0      1
      0      1      1      0      0      1
      1      0      0      0      0      0
      1      1      1      0      0      0
      0      1      0      0      0      1
      0      1      0      1      0      0
      1      0      1      1      0      0
      0      1      0      1      0      0
      0      0      1      0      0      0
      1      0      0      0      0      1
      1      0      0      1      1      0
      1      0      1      0      0      1
      0      1      0      1      0      1
      1      0      1      0      0      0
      1      1      1      0      0      0
      1      1      1      1      0      0
      1      0      0      0      0      0
      0      0      1      1      0      0
      0      0      1      1      1      0
      1      1      0      0      1      1
      1      0      0      1      0      0
      0      0      0      1      0      0
      0      0      1      0      1      1
      0      1      1      1      0      0
      1      0      1      1      0      0
      0      0      1      1      0      0
      1      1      0      0      0      1
      1      1      1      0      1      0
      1      1      1      1      1      0
      0      1      1      1      0      1
      1      1      0      1      0      0
      1      0      0      0      1      1
      0      1      1      0      1      0
      1      1      1      0      0      0
      1      0      1      0      1      0
      1      1      1      1      0      1
      1      0      0      1      1      0
      0      1      1      1      0      0
      0      0      1      0      1      1
      1      0      0      0      1      0
```

APPENDIX B: Multiplexed message 8SPK

```
:.Multiplexed_message_8PSK:
 Columns 1 through 6

  0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i

 Columns 7 through 12

  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
```

Columns 13 through 18

0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.0000 + 1.0000i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i
0.0000 + 1.0000i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.0000 + 1.0000i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i

Columns 19 through 24

0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i
0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i	0.7071 + 0.7071i
0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.0000 + 1.0000i	0.7071 + 0.7071i	0.0000 + 1.0000i

```
 Columns 25 through 30

  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i

Columns 31 through 36

  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i
```

```
Columns 37 through 42

  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i

Columns 43 through 48

  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i   0.0000 + 1.0000i   0.7071 + 0.7071i

Columns 49 through 50

  0.0000 + 1.0000i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i
  0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.0000 + 1.0000i   0.0000 + 1.0000i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.7071 + 0.7071i
  0.7071 + 0.7071i   0.0000 + 1.0000i
```

APPENDIX C: MATLAB code

```
close all
clear all
clc

% First Stage - The spreading of user messages via NOMA.
codeLength = 5;
numUsers = 6;
messageLength = 50;

spreadingCodes = randi([0,1],codeLength,numUsers);
%spreadingCodes = randi([0,1], codeLength, numUsers) * 2 - 1;
%spreadingCodes(spreadingCodes ==0)= -1;

messages = randi([0,1],messageLength,numUsers);

User_1codes = spreadingCodes(1,:);
User_2codes = spreadingCodes(2,:);
User_3codes = spreadingCodes(3,:);
User_4codes = spreadingCodes(4,:);
User_5codes = spreadingCodes(5,:);

User_1mes = messages(:,1)';
User_2mes = messages(:,2)';
User_3mes = messages(:,3)';
User_4mes = messages(:,4)';
User_5mes = messages(:,5)';

%Spreading occurs at the user device i.e., mobile smartphone.
% as a multplication of the codes and message.
for i = 1:length(User_1codes)
   for j = 1:length(User_1mes)
User_1spm(i,j) = User_1codes(i).*User_1mes(j);
User_2spm(i,j) = User_2codes(i).*User_2mes(j);
User_3spm(i,j) = User_3codes(i).*User_3mes(j);
User_4spm(i,j) = User_4codes(i).*User_4mes(j);
User_5spm(i,j) = User_5codes(i).*User_5mes(j);
   end
end
User_1spm;
User_2spm;
User_3spm;
User_4spm;
User_5spm;

%Profile 1 - All messages are 8PSK Modulated
PSK_Modulated_Message_User_1 = pskmod(User_1spm,8,pi/4);
PSK_Modulated_Message_User_2 = pskmod(User_2spm,8,pi/4);
PSK_Modulated_Message_User_3 = pskmod(User_3spm,8,pi/4);
PSK_Modulated_Message_User_4 = pskmod(User_4spm,8,pi/4);
PSK_Modulated_Message_User_5 = pskmod(User_5spm,8,pi/4);
multiplexed_message_8PSK = [PSK_Modulated_Message_User_1 ;
PSK_Modulated_Message_User_2;...
```

```
PSK_Modulated_Message_User_3;PSK_Modulated_Message_User_4;PSK_Modulated_Mes
sage_User_5];
%Profile 1 - All messages are 8PSK Modulated

%Profile 2 - Some messages are PSK Modulated in 4 PSK, 8PSK and others in
%16 PSK;
User_1spm_8PSK = pskmod(User_1spm,8,pi/4);
User_2spm_8PSK = pskmod(User_2spm, 8,pi/4);
User_3spm_16PSK = pskmod(User_3spm,16,pi/4);
User_4spm_8PSK = pskmod(User_4spm, 8,pi/4);
User_5spm_16PSK = pskmod(User_5spm,16,pi/4);
multiplexed_message_PSK_various = [User_1spm_8PSK ; User_2spm_8PSK;
User_3spm_16PSK;User_4spm_8PSK;User_5spm_16PSK];
QAM_Modulated_Message_User_1 = qammod(User_1spm,8);
QAM_Modulated_Message_User_2 = qammod(User_2spm,8);
QAM_Modulated_Message_User_3 = qammod(User_3spm,8);
QAM_Modulated_Message_User_4 = qammod(User_4spm,8);
QAM_Modulated_Message_User_5 = qammod(User_5spm,8);
multiplexed_message_QAM = [QAM_Modulated_Message_User_1 ;
QAM_Modulated_Message_User_2;...

QAM_Modulated_Message_User_3;QAM_Modulated_Message_User_4;QAM_Modulated_
Message_User_5];

% It is assumed that all messages are PSK modulated
% % setup of the MIMO Channel
% mimo_channel = comm.MIMOChannel
mimochannel = comm.MIMOChannel('SampleRate',1000,'PathDelays',[0 2e-3],
'AveragePathGains',[0 -5],...

'MaximumDopplerShift',5,'SpatialCorrelationSpecification','None','NumTransmitAntennas',50
, 'NumReceiveAntennas',4);
% Multiplexed modulated message is transmitted through the MIMO Channel.
Transmitted_Multiplexed_Message_1 = multiplexed_message_8PSK;
Transmitted_Multiplexed_Message_2 = multiplexed_message_PSK_various;
Transmitted_Multiplexed_Message_3 = multiplexed_message_QAM;

Receieved_Message_1 = mimochannel(Transmitted_Multiplexed_Message_1);
Receieved_Message_2 = mimochannel(Transmitted_Multiplexed_Message_2);
Receieved_Message_3 = mimochannel(Transmitted_Multiplexed_Message_3);

%Each of the received messages i.e., Received_Message_1, Received_Message_2,
% and Received_Message_3 constitutes the training input.
% Presentation of the training outputs.
% for Received_Message_1
Training_Output_1 = Receieved_Message_1;
Training_Input_1 = multiplexed_message_8PSK;
Training_Input_2 = Receieved_Message_2;
Training_Output_2 =multiplexed_message_PSK_various;
Training_Input_3 = Receieved_Message_3;
Training_Output_3 = multiplexed_message_QAM;
```

```
% Development of the Neural Network - Deep Learning Neural Network.
DNN_MUD = network;
Num_Inputs = 1;
DNN_MUD.numInputs = Num_Inputs;
x = 3;
DNN_MUD.numLayers = x; % One input layer, One Output layer, One Hidden layer
DNN_MUD.layers{1}.transferFcn = 'tansig';
DNN_MUD.layers{2}.transferFcn = 'tansig';
DNN_MUD.layers{3}.transferFcn = 'logsig';
DNN_MUD.layers{1}.size =60;
DNN_MUD.layers{2}.size = 40;
DNN_MUD.layers{3}.size = 8;

for i = 1:Num_Inputs
 DNN_MUD.inputConnect(1,i) = 1;
end

DNN_MUD.layerConnect(1,2) = 1;
DNN_MUD.layerConnect(2,3) = 1;
DNN_MUD.outputConnect(3) = 1;
%Optional - The inclusion of bias in the layers
DNN_MUD.biasConnect(1) = 1;
DNN_MUD.biasConnect(2) = 1;
DNN_MUD.biasConnect(3) = 1;
DNN_MUD.layers{1}.initFcn ='initnw';
DNN_MUD.layers{2}.initFcn ='initnw';
DNN_MUD.layers{3}.initFcn ='initnw';
%No delays in the layers.
DNN_MUD.trainFcn = 'trainlm';
%DNN_MUD = init(DNN_MUD); % Initialize the neural network.

Training_Output_1a = real(Training_Output_1)';
Training_Output_1b = imag(Training_Output_1)';
Training_Output_1 = [Training_Output_1a; Training_Output_1b];
%Training_Input_1 =
reshape(Training_Input_1,1,length(Training_Input_1(1,:)*length(Training_Input_1(:,1))));

Training_Input_1 = (Training_Input_1);
Training_Input_1(376:480) = ones(1,105);
Training_Input_1a = real(Training_Input_1)';
Training_Input_1a_grp_1 = Training_Input_1a(1:30);
Training_Input_1a_grp_2 = Training_Input_1a(31:60);
Training_Input_1a_grp_3 = Training_Input_1a(61:90);
Training_Input_1a_grp_4 = Training_Input_1a(91:120);
Training_Input_1a_grp_5 = Training_Input_1a(121:150);
Training_Input_1a_grp_6 = Training_Input_1a(151:180);
Training_Input_1a_grp_7 = Training_Input_1a(181:210);
Training_Input_1a_grp_8 = Training_Input_1a(211:240);
Training_Input_1a_grp_9 = Training_Input_1a(241:270);
Training_Input_1a_grp_10 = Training_Input_1a(271:300);
Training_Input_1a_grp_11 = Training_Input_1a(301:330);
Training_Input_1a_grp_12 = Training_Input_1a(331:360);
```

```
Training_Input_1a_grp_13 = Training_Input_1a(361:390);


Training_Input_1b = imag(Training_Input_1)';
Training_Input_1 = [Training_Input_1a_grp_1  Training_Input_1a_grp_2
Training_Input_1a_grp_3,...
    Training_Input_1a_grp_4 Training_Input_1a_grp_5 Training_Input_1a_grp_6
Training_Input_1a_grp_7,...
    Training_Input_1a_grp_8 Training_Input_1a_grp_9 Training_Input_1a_grp_10
Training_Input_1a_grp_11 Training_Input_1a_grp_12 ,...
    Training_Input_1a_grp_13];
Training_Input_1 = Training_Input_1(1:240);

Training_Input_1 = reshape(Training_Input_1,8,30);

% Assign labels or identifiers to each set of training data
labels = ones(1, size(Training_Output_1, 2)) * 1;

% Shuffle the data
idx = randperm(size(Training_Input_1, 2));
Training_Input_1 = Training_Input_1(:, idx);
Training_Output_1 = Training_Output_1(:, idx);
labels = labels(idx);


% Define training options
options = trainingOptions('adam', ...
   'MaxEpochs', 10, ...
   'MiniBatchSize', 128, ...
   'Verbose', true, ...
   'Plots', 'training-progress');

% Train the DNN with switching between different sets of training data
numMiniBatches = ceil(size(Training_Input_1, 2) / options.MiniBatchSize);
        epoch = 1:options.MaxEpochs;
        miniBatch = 1:numMiniBatches;
      startIndex = (miniBatch - 1) * options.MiniBatchSize + 1;
      endIndex = min(miniBatch * options.MiniBatchSize, size(Training_Input_1, 2));

      miniBatchInput = Training_Input_1(:, startIndex:endIndex);
      miniBatchOutput = Training_Output_1(:, startIndex:endIndex);
      miniBatchLabels = labels(startIndex:endIndex);


% Compute SVD for each matrix
[U_8PSK, S_8PSK, V_8PSK] = svd(multiplexed_message_8PSK);
SVD_8PSK = svd(multiplexed_message_8PSK);

% Extract singular values
singular_values_8PSK = diag(S_8PSK);
```

```
% singular values for training input
Training_Input_1 = (singular_values_8PSK);

% Normalize the training input
Training_Input_1 = Training_Input_1 / max(Training_Input_1(:));

% Connect the modulated messages for training output
Training_Output_1 = real(Training_Output_1)';

% Normalize the training output
Training_Output_1 = Training_Output_1 / max(Training_Output_1(:));


% Define the Deep Neural Network architecture
%layers = [
   %fullyConnectedLayer(64)...
   %reluLayer...
   %fullyConnectedLayer(32)...
   %reluLayer...
   %fullyConnectedLayer(numTransmitAntennas * numUsers)];

% Initialize and configure the neural network
DNN_MUD = feedforwardnet([64 32], 'trainlm'); % Create the network with two hidden
layers
DNN_MUD = init(DNN_MUD); % Initialize the neural network


% Train the DNN
%DNN_MUD = train(DNN_MUD, Training_Input_1.', Training_Output_1.', options);
%DNN_MUD = train(DNN_MUD,miniBatchInput,miniBatchOutput,'useParallel','yes');

% Perform multi-user detection using the trained DNN
%estimatedSymbols_DNN = predict(DNN_MUD, Training_Input_1.');


%%Display%%

%Display spreading codes
disp('1.Generated Spreading Codes:');
disp(spreadingCodes);

%Display messages
disp('2.Generated Random Messages:');
disp(messages);

% Display Multiplexed_message_8PSK matrices
disp('3.Multiplexed_message_8PSK:');
disp(multiplexed_message_8PSK);

% Display information about SVD computation
disp('4.SVD computation:');
disp('4.1. Dimensions of multiplexed_message_8PSK:');
```

```
disp(num2str(size(multiplexed_message_8PSK)));
disp('4.2. singular_values_8PSK:');
disp(singular_values_8PSK);

% Display information about multiplexed messages
disp('5.Multiplexed_message_8PSK_dimensions:');
disp(num2str(size(multiplexed_message_8PSK)));

%Display Training_Input_1
disp('6.Training_Input_1:');
disp(Training_Input_1);

% Diplay Training_Output_1
disp('7.Training_Output_1:')
disp(Training_Output_1);

 % Display training progress
 disp('8.Training progress:')
   fprintf('Epoch %d/%d, Mini-batch %d/%d\n', epoch, options.MaxEpochs, miniBatch,
numMiniBatches);


%Display DNN_MUD information
disp('9.DNN_MUD:');
disp(DNN_MUD);

%Display Mimochannel
disp('10.Mimochannel:');
disp(mimochannel);

%Display options info
disp('11. Options:');
disp(options);


 DNN_MUD.trainParam.max_fail = 3e4;
 DNN_MUD.trainParam.epochs = 1e35;
 DNN_MUD.trainParam.goal = 1e-6;
 DNN_MUD.trainParam.lr = 0.167;
 DNN_MUD.performFcn = 'mse';
 %DNN_MUD.plotFcns;
 DNN_MUD.trainParam.min_grad = 2e-15;
 DNN_MUD.divideFcn ='dividerand';
 DNN_MUD.divideParam.trainRatio = 0.7;
 DNN_MUD.divideParam.valRatio = 0.15;
 DNN_MUD.divideParam.testRatio = 0.15;

 [DNN_MUD, Training_Record] =
train(DNN_MUD,Training_Input_1.',Training_Output_1.');
 training_mse = Training_Record.perf;
 validation_mse = Training_Record.vperf;
 testing_mse = Training_Record.tperf;


    figure(1)
    h1 = plot(1:length(training_mse),training_mse,'o-','Markersize',12,'linewidth',1.5);
    hold on
    h2 = plot(1:length(training_mse),validation_mse,'s-','Markersize',12,'linewidth',1.5);
    hold on
    h3 = plot(1:length(training_mse),testing_mse,'o-','Markersize',12,'linewidth',1.5);
    hold on
    ylabel('Mean Square Error: MUD-NN','FontSize',12,'FontWeight','bold');
    xlabel('Training Epoch','FontSize',12,'FontWeight','bold');
    h4 = legend([h1 h2 h3],'Training','Validation','Testing');
    set(h4,'fontsize',12,'fontweight','bold');
    set(gca,'fontsize',12,'fontweight','bold');
    plotperf(Training_Record)

    view(DNN_MUD)

   END
```