UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

CONSTRUCTING AND DEPLOYING AI NATIVE PHYSICAL LAYER USING WIRELESS AUTOENCODERS FOR ENHANCED RF AND OPTICAL WIRELESS COMMUNICATION

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

Amer Zayegh
Norman, Oklahoma
2025

CONSTRUCTING AND DEPLOYING AI NATIVE PHYSICAL LAYER USING WIRELESS AUTOENCODERS FOR ENHANCED RF AND OPTICAL WIRELESS COMMUNICATION

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Chad B. Roller, Chair

Dr. Hazem H. Refai, Co-Chair

Dr. Daniel E. Hamlin

Dr. Samuel Cheng

Dr. Choon Yik Tang

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, **Prof. Hazem H. Refai**, for his unwavering support, insightful guidance, and continuous encouragement throughout my academic journey. Their mentorship and expertise have profoundly shaped the direction and quality of this dissertation, and I am sincerely thankful for the opportunity to learn under their supervision.

I would also like to sincerely thank my chair, **Dr. Chad B. Roller**, whose support helped me bring this work to completion.

I also wish to thank the faculty and staff of the **Department of Electrical and Computer Engineering** at **University of Oklahoma** for fostering an environment of academic excellence and for providing the tools and knowledge that have enriched my research. I am especially grateful to **Committee Members** for their constructive feedback, thoughtful discussions, and valuable perspectives that strengthened this work.

My heartfelt appreciation goes to **Ms. Denise Davis,** ECE Administrative Support Specialist, whose support and empathy have made this journey rewarding.

Most importantly, I would like to express my deepest love and gratitude to my family. To my parents and siblings, thank you for your sacrifices, your unwavering belief in me, and for instilling in me the values of perseverance and curiosity. To my wife (**Lama Kazzara**), your patience, support, and unconditional encouragement carried me through every challenge and inspired me to keep going. To my children (**Ammar and Laith**), your joy reminded me every day of what truly matters. Your belief in me has been the foundation of everything I have achieved.

To all those who supported me along the way—academically, emotionally, or personally—thank you. This accomplishment is as much yours as it is mine.

Contents

Contents v
List of Figures viii
List of Tables x
Abstract xi
1. Introduction 1
1.1 Background and Motivation 1
1.2 Research Problem 2
1.3 Objectives and Scope 2
1.4 Limitations of Conventional Decoders 3
1.5 Deep Learning for Wireless and Optical Decoding 4
1.6 Dynamic Neural Network Architectures 4
1.7 Contributions of the Dissertation 5
1.8 Dissertation Structure 5
2 Deep Learning for Wireless Communications 8
2.1 Introduction 8
2.2 Related Work 8
2.3 Challenges in DL-based wireless communication 22
3 Autoencoder Principles, Types, and Applications 23
3.1 Introduction 23
3.2 Autoencoder Vs Conventional Encoder-Decoder 24
3.3 Autoencoder Structure and Work Principles 24
3.4 Autoencoder Hyperparameters 28
3.5 Autoencoder Types and Applications: 28
3.6 Autoencoder for wireless communication 31
4 Deep Learning Decoders in Optical Wireless Communication Operating under Atmospheric Challenges 34
4.1 Introduction 34
4.2 Related Work 35
4.3 Experimental Setup 37
4.3.1 Channel Sounding 39

4.3.2 Atmospheric Turbulence 41
4.4 Methodology 44
4.5 RESULTS AND DISCUSSION 47
4.6 Conclusion 51
5 Simulation, Implementation, and Evaluation of a Deep Learning-Based Wireless Decoder Using Real-World Over-the-Air Data 53
5.1 Background 54
5.2 Simulation of OFDM-based wireless autoencoder 55
5.2.1 OFDM System Setup 55
5.2.2 Autoencoder Architecture and Training 55
5.2.3 Simulation Results 57
5.3 Implementation and Evaluation of a Deep Learning-Based Wireless Decoder Using Real-World Over-the-Air Data 59
5.3.1 Experimental Setup 60
5.4 Methodology 63
5.4.1 RESULTS AND DISCUSSION 65
5.5 Conclusion 69
6 Toward AI Native Wireless Communication Physical Layer Using Dynamic Deep Neural Networks 71
6.1 Introduction 71
6.2 Dynamic Neural Networks 72
6.3 Dynamic DL-Based Decoder Architecture: 73
6.3.1 Modulation-aware network switching 74
6.3.2 Early Exit Architecture 75
6.3.3 Dynamic Parameters 77
6.3.4 Experimental Setup 78
6.3.5 Methodology 80
6.3.6 Architecture Overview 80
6.3.7 Training Strategies for Each Dynamic Inference Method 82
6.3.8 Modulation Classifier Design 84
6.3.9 Results and Discussion: 86
6.4 Dynamic DL-Based Encoder: 92

6.4.1 Experimental Setup for Encoder Training 94
6.5 Adaptive Neural Transceiver Design: Integrating Dynamic Encoder and Decoder in a Practical Autoencoder Framework 98
6.5.1 System Overview 99
6.5.2 Fine-Tuning Strategy 100
6.6 Results and Discussion: 102
6.6.1 BLER Performance Comparison 107
6.6.2 Modulation-Specific Observations 108
7 Conclusion, Open Research Challenges 110
7.1 Limitations and Challenges 113
7.2 Open Research Topics 114
7.3 Final Remarks 114
References 115

List of Figures

FIGURE 2-1. PERFORMANCE OF DL VS. CONVENTIONAL APPROACHES FOR RIS [1] 10
FIGURE 2-2. DL-BASED CHANNEL STATE INFORMATION COMPRESSION [3]. 11
FIGURE 2-3. BIT ERROR RATE VS SNR FOR DL-BASED SIGNAL DETECTION AND RECONSTRUCTION [8] 11
FIGURE 2-4. ENERGY CONSUMPTION IN KWH FOR CELL [9] 12
FIGURE 2-5. SIMULATION RESULTS OF LOAD BALANCING ALGORITHMS [9] 13
FIGURE 2-6. LOSS OF GANS CHANNEL ESTIMATION [10] 14
FIGURE 2-7. CONFUSION MATRIX OF RF CLASSIFIER PROPOSED IN [11] 15
FIGURE 2-8. WIRELESS AUTOENCODER OVER AWGN CHANNEL PROPOSED IN [13] 16
FIGURE 2-9. WIRELESS AE PERFORMANCE IN BPSK [13] 17
FIGURE 2-10. MULTIPLE USERS WIRELESS COMMUNICATION SYSTEM [13] 17
FIGURE 2-11. BLER PERFORMANCE OF MULTIPLE USERS WIRELESS AE [13] 18
FIGURE 2-12. BLOCK DIAGRAM OF AE-OFDM SYSTEMS WITH GAN AND INTERFERENCE SUPPRESSION [15] 18
FIGURE 2-13. BER RESULTS OF AE-OFDM SYSTEM WITHOUT INTERFERENCE [15] 19
FIGURE 2-14. BER RESULTS OF AE-OFDM WITH 10 DEGREES PHASE OFFSET AND 30 HZ FREQUENCY OFFSET [15] 19
FIGURE 2-15. AE VERSUS CONVENTIONAL CONSTELLATIONS AT 30DB SNR [15] 20
FIGURE 2-16. AE VERSUS CONVENTIONAL CONSTELLATIONS AT -5DB SNR [15] 20
FIGURE 2-17. DATA AUGMENTATION EFFECT ON AE PERFORMANCE (100 SAMPLES ORIGINAL DATASET) [16] 21
FIGURE 2-18. DATA AUGMENTATION EFFECT ON AE PERFORMANCE (1000 SAMPLES ORIGINAL DATASET) [16] 21
FIGURE 3-1. AUTOENCODER STRUCTURE 25
FIGURE 3-2. AUTOENCODER LAYERS 26
FIGURE 3-3. AUTOENCODER HIDDEN LAYERS AND STATES 27
FIGURE 3-4. GRAPH REPRESENTATION OF A PROBABILISTIC MODEL OF AUTOENCODER [21] (MODIFIED) 27
FIGURE 3-5. PCA VERSUS AUTOENCODER DIMENSIONALITY REDUCTION [27] 29
FIGURE 3-6. SPARSE AUTOENCODER [27] 30
FIGURE 3-7. WORKING PRINCIPLE OF CONTRACTIVE AUTOENCODERS [27] 30
FIGURE 3-8. AUTOENCODER RECEIVER WITH RTN [17] 32
FIGURE 3-9. GAN-BASED WIRELESS AUTOENCODER [31] 32
FIGURE 4-1 EXPERIMENTAL SETUP FOR WEATHER TURBULENCE OF OWC LINK 38
FIGURE 4-2 LAB EXPERIMENTAL SETUP FOR DATA COLLECTION 38
FIGURE 4-3 RECEIVED DATA WAVEFORMS. 39
FIGURE 4-4 SAMPLE OF CHANNEL-SOUNDING PULSES AND CHANNEL POWER PROFILE 40
FIGURE 4-5 CHANNEL ATTENUATION AS A FUNCTION OF FOG DENSITY AND TEMPERATURE 41
FIGURE 4-6 EYE DIAGRAM (A) UNDER LIGHT FOG, (B) UNDER MODERATE FOG, (C) UNDER THICK FOG 42
FIGURE 4-7 RECEIVED DATA HISTOGRAM WITH BEST-FIT LOGNORMAL DISTRIBUTION (A) UNDER LIGHT FOG, (B) UNDER MODERATE FOG, (C) UNDER THICK FOG 44
FIGURE 4-8 FULLY CONNECTED FEED-FORWARD NEURAL NETWORK (NN) STRUCTURE 44
FIGURE 4-9 TRAINING LOSS FOR DIFFERENT LEARNING RATES 46
FIGURE 4-10 DL-DECODER FOR LIGHT AND MODERATE FOG 48
FIGURE 4-11 BLER VS $Eb \ /N0$ FOR LIGHT AND MODERATE FOG 48
FIGURE 4-12 VALIDATION LOSS AND ACCURACY FOR LIGHT AND MODERATE FOG 49
FIGURE 4-13 BLER VS $Eb/N0$ AND TRAINING PERFORMANCE FOR THICK FOG (SHALLOW STRUCTURE) 50
FIGURE 4-14 DL-DECODER FOR THICK FOG 50
FIGURE 4-15 BLER VS $Eb/N0$ AND TRAINING PERFORMANCE FOR THICK FOG (MODIFIED STRUCTURE) 50
FIGURE 4-16 BLER AND FLOPS FOR DIFFERENT NUMBERS OF DEEP LAYERS 51
FIGURE 5-1 CONVENTIONAL OFDM TRANSCEIVER 53
FIGURE 5-2 AUTOENCODER STRUCTURE FOR OFDM SIMULATION 56

FIGURE 5-3 BLER VERSUS EB/N0 FOR QPSK-OFDM BASED AUTOENCODER. 57
FIGURE 5-4 VALIDATION LOSS AND ACCURACY VERSUS TRAINING ITERATIONS (QPSK-AE) 58
FIGURE 5-5 BLER VERSUS EB/N0 FOR 16-QAM-OFDM BASED AUTOENCODER. 58
FIGURE 5-6 VALIDATION LOSS AND ACCURACY VERSUS TRAINING ITERATIONS (16-QAM-AE) 59
FIGURE 5-7 SDR TRANSMITTER AND RECEIVER 60
FIGURE 5-8 OFDM BLOCK DIAGRAM WITH AUTOENCODER RECEIVER 60
FIGURE 5-9 EXPERIMENTAL SETUP 61
FIGURE 5-10 POWER HISTOGRAM OF LAB ENVIRONMENT SURVEY 62
FIGURE 5-11 POWER HISTOGRAM IN PRESENCE OF TX SIGNAL 62
FIGURE 5-12 FULLY CONNECTED FEED-FORWARD NEURAL NETWORK (NN) STRUCTURE 63
FIGURE 5-13 TRAINING LOSS FOR DIFFERENT LEARNING RATES. 65
FIGURE 5-14 DL-DECODER STRUCTURE 66
FIGURE 5-15 BLER PERFORMANCE OF DL-RECEIVER FOR QPSK 66
FIGURE 5-16 VALIDATION LOSS AND ACCURACY FOR QPSK DL-BASED DECODER 67
FIGURE 5-17 BLER PERFORMANCE OF DL-RECEIVER FOR 16-QAM 67
FIGURE 5-18 VALIDATION LOSS AND ACCURACY FOR 16-QAM DL-BASED DECODER 68
FIGURE 5-19 BLER FOR DIFFERENT NUMBER OF HIDDEN LAYERS IN QPSK DL-DECODER 69
FIGURE 5-20 FLOPS FOR DIFFERENT NUMBERS OF HIDDEN LAYERS 69
FIGURE 6-1 MODULATION-AWARE NETWORK SWITCHING 74
FIGURE 6-2 EARLY EXIT ARCHITECTURE 76
FIGURE 6-3 DYNAMIC PARAMETERS ARCHITECTURE 77
FIGURE 6-4 EXPERIMENTAL TESTBED FOR DYNAMIC NN DECODER EVALUATION 79
FIGURE 6-5 LEARNING RATE VS EPOCHS 82
FIGURE 6-6 MODULATION CLASSIFIER CONFUSION MATRIX 86
FIGURE 6-7 NETWORK SWITCHING DECODER STRUCTURE 87
FIGURE 6-8 BLER PERFORMANCE OF NETWORK SWITCHING DECODER 88
FIGURE 6-9 EARLY EXIT DECODER STRUCTURE 88
FIGURE 6-10 BLER PERFORMANCE OF EARLY EXIT DECODER 89
FIGURE 6-11 DYNAMIC PARAMETERS RELOADING STRUCTURE 89
FIGURE 6-12 BLER PERFORMANCE OF PARAMETERS RELOADING DECODER 90
FIGURE 6-13 DYNAMIC ENCODER USING NETWORK SWITCHING 93
FIGURE 6-14 DIAGRAM OF CONSTELLATION GENERATION AND ENCODER TRAINING 94
FIGURE 6-15 EVM ILLUSTRATION [71] 95
FIGURE 6-16 IDEAL AND DL-BASED GENERATED CONSTELLATION POINTS FOR QPSK 96
FIGURE 6-17 EVM VALUES FOR DL-BASED QPSK ENCODER 96
FIGURE 6-18 EVM CONVERGENCE VS. TRAINING EPOCHS 96
FIGURE 6-19 EVM VALUES (A) DL-BASED 16-QAM ENCODER (B) DL-BASED 64-QAM ENCODER 97
FIGURE 6-20 END-TO-END DYNAMIC WIRELESS SDR-AUTOENCODER (NETWORK SWITCHING DECODER) 103
FIGURE 6-21 BLER PERFORMANCE FOR WIRELESS AE (NETWORK SWITCHING DECODER) 103
FIGURE 6-22 END-TO-END DYNAMIC WIRELESS SDR-AUTOENCODER (PARAMETER ADAPTION DECODER) 104
FIGURE 6-23 BLER PERFORMANCE FOR WIRELESS AE (PARAMETER ADAPTION DECODER) 105
FIGURE 6-24 END-TO-END DYNAMIC WIRELESS SDR-AUTOENCODER (EARLY EXIT DECODER) 106
FIGURE 6-25 BLER PERFORMANCE FOR WIRELESS AE (EARLY EXIT DECODER) 107

List of Tables

TABLE 5-1 OFDM SYSTEM PARAMETERS.....55
TABLE 6-1 COMPLEXITY AND ENERGY CONSUMPTION OF DYNAMIC NN STRUCTURES.....91
TABLE 6-2 EFFICIENCY AND ARCHITECTURAL TRADE-OFFS 108

Abstract

Conventional wireless communication systems rely on fixed, block-based signal processing pipelines that are manually optimized and lack the ability to adapt in real time to dynamic channel conditions, hardware impairments, or varying modulation requirements. These limitations make them increasingly inefficient in modern communication environments characterized by mobility, interference, and diverse quality-of-service demands. As wireless standards like IEEE 802.11 and beyond continue to evolve, there is a growing need for intelligent transceivers that can adapt their complexity, structure, and behavior on demand. This dissertation addresses that need by introducing a deep learning-based framework that replaces static transceiver components with dynamic neural networks capable of learning, adapting, and co-optimizing over the air.

This dissertation presents a unified deep learning framework for designing intelligent, modulation-adaptive wireless communication systems. Motivated by the limitations of conventional block-based architectures in coping with dynamic channel conditions and diverse modulation schemes, this work proposes a novel end-to-end system architecture composed of dynamic neural networks (DNNs) at both the transmitter and receiver. The study begins by developing multiple decoder architectures capable of runtime structural adaptation, including modulation-aware network switching, dynamic weight reloading, and early exit inference. Each decoder strategy is tailored for QPSK, 16-QAM, and 64-QAM, enabling complexity-efficient decoding while improving block error rate (BLER) under varying SNR conditions.

The dissertation then introduces a dynamic neural encoder based on modulation-aware network switching. A single architecture houses specialized subnetworks for each modulation order, and conventional channel state information (CSI) is used to select the appropriate path at runtime. This encoder is trained using constellation points generated by a standard IEEE 802.11n stack via a vector signal generator, with performance evaluated in terms of error vector magnitude (EVM).

The encoder and decoder are subsequently integrated into a full end-to-end autoencoder-based communication system. Fine-tuning is applied to jointly optimize the encoder and decoder using both simulated and real over-the-air data collected via a USRP X310 software-defined radio.

The final system is evaluated comprehensively across multiple metrics—BLER, EVM, floating-point operations (FLOPs), and energy consumption—demonstrating significant gains over conventional systems. Among the decoder strategies, dynamic weight reloading achieves the best overall performance, while early exit provides the most efficient inference for low-order modulations.

This work demonstrates the feasibility and effectiveness of dynamic deep learning architectures for next-generation physical layer design. By leveraging runtime adaptability, structural modularity, and joint optimization, the proposed system offers improved robustness, efficiency, and scalability. The findings lay the groundwork for intelligent, resource-aware transceivers capable of operating effectively across a wide range of wireless environments.

1. Introduction

1.1 Background and Motivation

The demand for high-speed, reliable, and adaptive communication systems has grown exponentially with the proliferation of mobile devices, the rise of data-intensive applications, and the expansion of wireless-enabled services in sectors such as healthcare, transportation, defense, and space exploration. As wireless communication becomes increasingly central to our digital infrastructure, new technologies are required to meet the stringent performance requirements of next-generation networks. Traditional radio-frequency (RF)-based systems, although mature, are being designed to accommodate higher spectral efficiency and resiliency. Meanwhile, Optical Wireless Communication (OWC) has emerged as a promising complementary technology, particularly in scenarios where RF communication is constrained by bandwidth, interference, or security concerns.

In both RF and OWC systems, signal degradation due to environmental conditions, hardware impairments, and channel variability remain major challenge. These issues are further exacerbated in real-world deployment scenarios where channel models often deviate from idealized assumptions. Overcoming these limitations demands not only robust physical layer techniques but also intelligent and adaptive receiver architectures capable of learning and generalizing on real-world observations.

The introduction of deep learning (DL) into wireless communication has signaled a paradigm shift. Deep Neural Networks (DNNs) are capable of approximating complex non-linear functions, learning from empirical data rather than relying solely on analytical or computer simulated models. These capabilities make them particularly well-suited for tasks such as channel decoding, symbol detection, demodulation, and joint source-channel decoding. However, the adoption of DL in real-world communication systems remains limited by two critical factors: (1) the heavy reliance on simulated datasets for training and evaluation, and (2) the computational burden associated with deploying static DNN architectures across dynamic wireless environments.

This dissertation addresses these limitations by developing and experimentally evaluating deep learning-based physical layer receivers for both RF and OWC systems using real-world datasets. Furthermore, it introduces and validates a novel class of **dynamic neural decoder architectures** that adapt inference complexity based on modulation schemes and channel conditions. The combination of empirical evaluation and adaptive design aims to advance the practical deployment of AI-enhanced communication systems.

1.2 Research Problem

Most existing DL-based communication systems are developed and validated through simulation environments. These simulations often make idealistic assumptions about the wireless channel, ignoring real-world phenomena such as multi-path fading, synchronization mismatches, analog front-end distortions, and external interference. Consequently, decoders trained under these conditions tend to underperform when deployed in physical systems.

Moreover, conventional DNN-based decoders are typically designed as static models, optimized for a fixed data training set collected under certain conditions. In adaptive communication environments where modulation orders and coding schemes vary dynamically, static neural networks suffer from a design mismatch. They either become over designed for low-complex tasks or under designed for high-complexity tasks. This mismatch leads to suboptimal performance or excessive energy consumption, particularly in resource-constrained devices.

Thus, the research problem addressed in this dissertation can be stated as follows:

"How can deep learning-based physical layer autoencoders be designed, trained, and deployed using real-world RF or optical wireless datasets, while maintaining adaptability, efficiency, and robustness in dynamic and imperfect channel conditions"

1.3 Objectives and Scope

The primary objectives of this dissertation are:

1. **To develop deep learning-based decoders** for RF and OWC systems and evaluate their performance using real experimental data collected in a lab setting under certain conditions.

2. **To investigate the limitations of model-based decoders** in the presence of non-idealities: hardware impairments, interference, and environmental distortions.
3. **To design dynamic neural network decoder architectures** that adapt their complexity based on modulation format and/or channel state information to improve energy efficiency without sacrificing accuracy.
4. **To experimentally validate these dynamic architectures** using both simulated and real datasets, demonstrating their suitability for deployment in real-time settings.

The scope of this dissertation is confined to the physical layer of communication systems, with a particular focus on the decoder component. Both RF and optical communication systems are considered, with real-world datasets collected using a custom-designed Software-Defined Radio (SDR) testbed and an optical weather chamber emulating fog and atmospheric turbulence.

1.4 Limitations of Conventional Decoders

Traditional decoding techniques are typically derived from theoretical models based on statistical assumptions about the channel. For instance, maximum-likelihood (ML) decoding, minimum mean square error (MMSE), or Zero Forcing (ZF) techniques assume specific noise distributions and fading behaviors. In practice, however, the channel response can be affected by:

- Hardware non-linearities (e.g., power amplifier distortion, I/Q imbalance)
- Synchronization offsets and sampling errors
- External interference from nearby transmitters or co-existing wireless technologies
- Environmental conditions such as fog, temperature gradients, and turbulence (particularly in OWC systems)

In such cases, model-based decoders trained or optimized for ideal conditions may yield poor performance. Furthermore, these conventional algorithms are not readily adaptable to changes in modulation or channel complexity without manual reconfiguration.

1.5 Deep Learning for Wireless and Optical Decoding

Deep learning offers a data-driven alternative to model-based decoding. By learning a statistical mapping from received signal samples to symbol probabilities or bit labels, neural decoders can capture effects that are otherwise hard to model explicitly. Several architectures have been explored in the literature, including:

- Fully connected networks for symbol detection
- Convolutional Neural Networks (CNNs) for time-series classification
- Recurrent Neural Networks (RNNs) and attention-based models for sequential decoding

These networks have shown promise in simulated environments but remain underexplored in realistic deployment scenarios. For instance, in OWC systems, atmospheric attenuation, turbulence, and scattering are difficult to model precisely, yet DL models can be trained to handle these conditions given sufficient real-world data.

This dissertation contributes to this area by designing and evaluating neural decoders on real datasets captured over RF channels (IEEE 802.11n) and through OWC links exposed to fog and turbulences generated in a laboratory chamber. These experiments serve to benchmark the practical performance of neural decoders and identify the limitations of conventional designs.

1.6 Dynamic Neural Network Architectures

To enable deployment in real-world, power-constrained communication systems, this dissertation proposes and evaluates several dynamic decoder architectures, including:

- **Modulation-aware selection networks**: Lightweight classifiers determine the modulation scheme and activate corresponding decoder sub-networks optimized for that scheme.
- **Early-exit networks**: Networks with multiple output layers allow decisions to be made early in the pipeline for simpler inputs, reducing computation.
- **Weight-switchable models**: A single shared decoder structure with dynamic weight loading, allowing flexibility while reducing memory requirements.

These architectures offer a mechanism to scale decoder complexity in real-time, aligning computational effort with task difficulty. Such designs are critical for edge applications like mobile devices, autonomous vehicles, and IoT networks where energy efficiency is a primary constraint.

1.7 Contributions of the Dissertation

The major contributions of this dissertation are summarized as follows:

1. **Empirical Dataset Collection**: A real-time SDR testbed and an optical weather chamber are used to collect high-fidelity RF and OWC datasets, incorporating channel impairments and environmental distortions.
2. **DL-Based Physical Layer Decoders**: Deep neural network decoders are designed, trained, and evaluated on the collected real datasets, demonstrating superior performance (between 0.25dB to 0.38dB on average) over conventional model-based approaches under challenging conditions.
3. **Dynamic Decoder Architectures**: Novel neural network structures are introduced that dynamically adapt inference complexity based on modulation or channel conditions, significantly reducing computational overhead without degrading accuracy.
4. **Comprehensive Evaluation**: The proposed models are assessed in terms of Block Error Rate (BLER), energy consumption, latency, and generalization ability across various channel and hardware scenarios.

1.8 Dissertation Structure

This dissertation is organized into six main chapters, each addressing a distinct aspect of the research on deep learning (DL)-based physical layers for RF and optical wireless communication (OWC) systems. The chapters are structured as follows:

- **Chapter 1: Introduction**

  This chapter presents the problem statement, motivation, background, objectives, and main contributions of the dissertation. It also outlines the structure and flow of the research work.

- **Chapter 2: Deep Learning for Wireless Communication**

  This chapter introduces key deep learning techniques and their relevance to wireless

communications. It discusses various DL architectures, explores current opportunities in the field, and highlights associated challenges.

- **Chapter 3: Autoencoder**

  This chapter presents the principles of autoencoders, including their types, models, and applications in communication systems. It lays the theoretical foundation for the autoencoder-based architectures used in subsequent chapters.

- **Chapter 4: Deep Learning for Optical Wireless Communication (DL for OWC)**

  This chapter focuses on the application of DL techniques to OWC systems. It includes a review of relevant literature, the channel model under fog and turbulence conditions, and details of the experimental setup. The chapter introduces a DL-based decoder, analyzing its structure, performance, and computational complexity in terms of FLOPs, processing time, and energy consumption.

- **Chapter 5: Deep Learning for Wireless Communication**

  This chapter explores DL-based physical layer implementations in conventional wireless systems using software-defined radios (SDRs). It covers the simulation and experimental realization of DL-based OFDM systems, including receiver and transmitter designs. Performance and complexity analyses are carried out for each component.

- **Chapter 6: Dynamic Neural Networks for DL-Based Physical Layer**

- This chapter introduces dynamic neural networks (DNNs) tailored for adaptive physical layer design. It covers their principles, structures, and applications, including dynamic DL-based IEEE 802.11n receivers and transmitters using SDR. A joint end-to-end dynamic wireless autoencoder is presented with blind and CSI-based adaptation. The chapter concludes with an explainable dynamic autoencoder, comparing neural inference with analytical models.

- **Conclusion and Future Work**

  This final chapter summarizes the key findings of the research and suggests future directions, including generalization to new wireless standards, real-time adaptation, and robust DL deployment.

- **References**

  A complete list of references cited throughout the dissertation.

2 Deep Learning for Wireless Communications

2.1 Introduction

Wireless communication forms the backbone of the increasingly connected world. To meet the ever-growing demand for faster, more reliable, and more efficient wireless communication systems, researchers and engineers are exploring the capabilities of various deep learning schemes to tackle some of the challenges in the field.

Deep learning (DL) offers a change in thinking about the design and optimization of wireless communication systems. In their design and optimization, conventional wireless communication systems rely on known channel model [1]. Deep learning algorithms utilize the power of neural networks to learn complex patterns from vast amounts of data, enabling them to figure out optimized solutions. These solutions adapt the communication system, in real time, to operate under varying channel conditions. This ability to learn from data, rather than relying on explicit models, makes deep learning particularly well-suited for addressing the inherent uncertainties and complexities of wireless communication channels.

This chapter explores the role of deep learning in wireless communication. Through multiple related works and practical examples, we demonstrate how DL can significantly enhance the performance and efficiency of wireless communication systems. Furthermore, we discuss the challenges of deep learning in wireless communication, and emerging technologies.

2.2 Related Work

In [1] researchers demonstrate various approaches to utilizing machine learning (ML) and DL in the optimization process of wireless communication systems. They depict the advantages of data-driven system design and optimization over the conventional approach. They emphasize the importance of selecting an appropriate neural network architecture to successfully approximate the targeted model. They confirmed that channel modeling is an essential process in conventional wireless communication systems. Using channel estimation, wireless communication systems can, approximately, model the impairments over the symbols being transmitted [1]. However, this process could be computationally expensive and susceptible to significant uncertainties due to

complex channel models. In addition, different channel natures can be modeled using different mathematical forms [1]. For example, the sparsity structure of mmWave Massive MIMO makes sparse path-based representations proper to model such systems in the spatial domain [1]. Nevertheless, for frequency-selective channels, time and frequency domain models are more appropriate to show the variant multi-path delays and their consequences on the frequency correlation [1]. In addition, minimizing the loss function (squared error in channel estimator) does not guarantee a global end-to-end optimization of the system design. Thus, researchers in wireless communication use their expertise to select an appropriate channel model and optimization function [1]. Authors confirmed that data-driven approaches can overcome explicit channel modeling by training the neural network (NN) using a considerably large number of channel samples [1]. Trained NN can directly map the problem space to the corresponding optimized solutions.

Later, the resulting channel model is used to optimize the performance of the system. Optimization process could be a non-convex problem that involves high-dimensional parameters [1]. Authors suggest that the ML/DL approach can outperform model-based approaches by offering “more efficient” solutions for such problems. On the other hand, they highlight the challenges in designing an optimal encoder/decoder for source, channel coding, and compression.

In terms of applications, researchers in [1] explored the capabilities of DL to maximize the capacity using reconfigurable intelligent surface (RIS) used for massive MIMO systems. A vast number of antenna arrays are used at both transmitter and receiver ends. These arrays are equipped with reflecting elements that can be “dynamically reconfigured” to allocate the beam toward desired users. This offers a higher channel capacity [2]. Given the sparse high-dimensional nature of the channel, channel estimation using uplink pilots is quite challenging. Furthermore, the optimization of RIS based on the channel model is a nonconvex problem. Authors investigated the ability of graph neural networks (GNN) to bypass the channel modeling by training a neural network on channel samples to produce the optimized coefficients. Results shown in Figure 2-1 illustrate that DL approaches outperform the conventional linear minimum mean-squared error

(LMMSE) method based on channel estimation and converge to the optimal performance much faster.

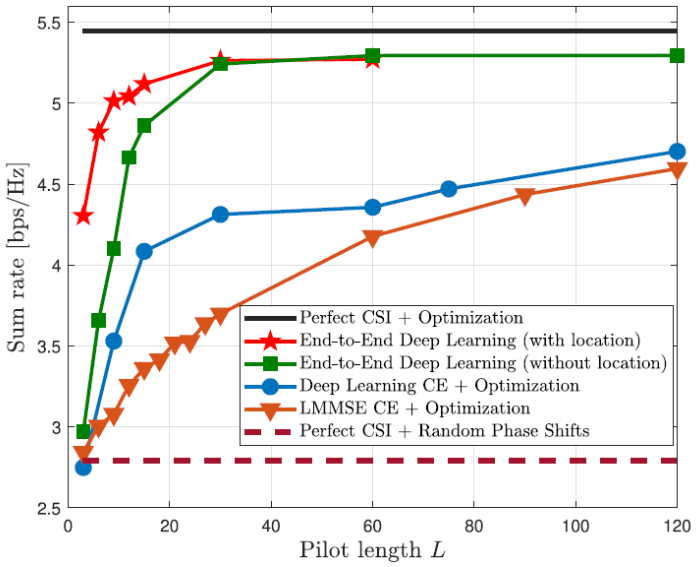


Figure 2-1. Performance of DL Vs. conventional approaches for RIS [1]

In [3] researchers investigated DL performance for intelligent signal compression. Traditional wireless communication systems suffer from excessive feedback overhead [4]. Compressive sensing (CS) is efficient in compressing the channel state information (CSI) at the user end and recovering it at the network end. However, CS efficiency depends on the accuracy of the channel estimation. Moreover, traditional recovery algorithms have slow convergence [5]. A deep convolutional network (DCN) is trained in [6] to apply the inverse transformation from compressed measurements of signals to speed up the recovery process. In [7] DL is utilized to develop a NN that learns to produce "near-optimal" codewords from CSI and recover the CSI from the corresponding codewords. Figure 2-2 depicts the structure of the CSI encoder and decoder proposed in [7]. The encoder uses the angular delay matrix of the channel as input to generate feature maps at the output of the convolutional layer. These maps are then vectorized and passed through a fully connected layer to output the compressed CSI. At the other end, the decoder learns to recover the CSI from the compressed representation. Results confirmed that the proposed structure outperforms the conventional approach.

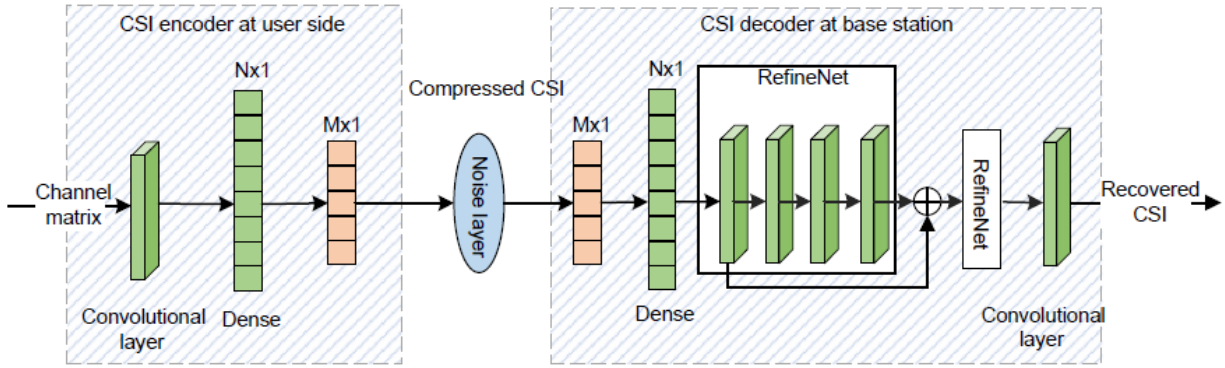


Figure 2-2. DL-based channel state information compression [3].

Researchers in [8] use simulation data to offline train a DL NN, consisting of five fully connected layers, to detect the signal and reconstruct the transmitted symbols. Simulation results in Figure 2-3 show that NN achieved a lower bit error rate than the traditional minimum mean-square error (MMSE) [8].

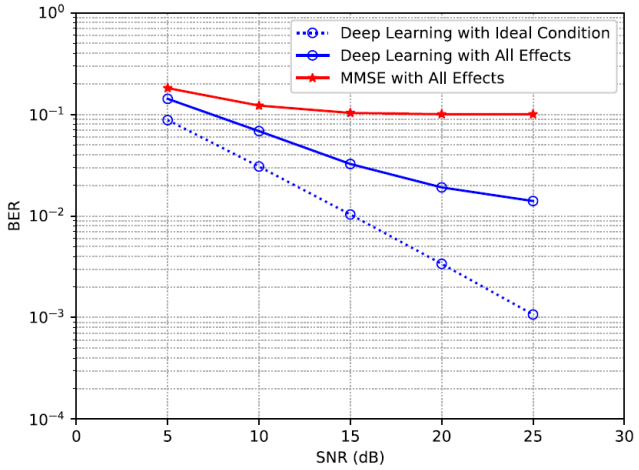


Figure 2-3. Bit Error Rate Vs SNR for DL-based signal detection and reconstruction [8]

In [9] researchers looked at NN nonlinear processing capabilities potential at the network level. The enormous number of base stations being deployed led to an increment in energy consumption, emissions, and operational overhead. Cell offloading based on traffic proposed in [10] was able to save energy by 15.4% on average [10]. “Switching off” the cell when the predicted traffic is less than a “fixed threshold” is applied at symbol, channel, and/or carrier levels [9]. However, data-driven approach for load prediction and threshold setting proposed in [9] (see Figure 2-4), enhanced the efficiency of energy saving to 63% on average [9].

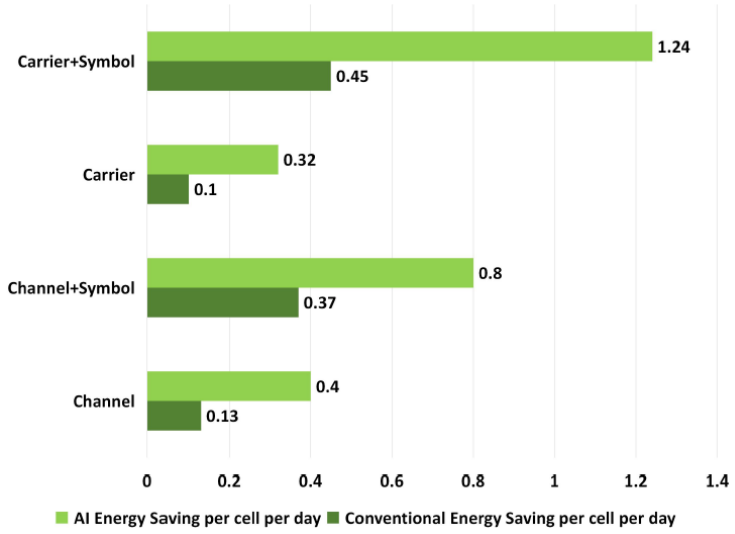


Figure 2-4. Energy consumption in KWh for cell [9]

In their comprehensive study, the authors in [9] emphasized the role of DL in both modulation recognition and wireless technology identification. They highlighted that the conventional spectrum sensing techniques and signal classification approaches have shown limitations in adaptability and performance [9]. These limitations prompted exploration of cognitive radio and dynamic spectrum access mechanisms.

Researchers classified modulation recognition methods into two classes: likelihood-based and feature-based methods. Likelihood based recognition depends on hypothesis testing and provide theoretically optimal performance. However, this approach is computationally intensive and might be impractical for real-time applications [9]. On the other hand, feature based method utilizes extracted features, such as instantaneous time parameters (e.g., amplitude, phase, frequency), statistical features, and constellation characteristics, which are then classified using traditional machine learning (ML) models or artificial neural networks. Nevertheless, this approach relies heavily on domain expertise and lack generalization across varying environments.

To overcome these drawbacks, the authors advocate for the use of deep learning models, particularly convolutional neural networks (CNNs) and long short-term memory (LSTM) networks, which learn discriminative features directly from raw signal data (e.g., I/Q samples or amplitude-phase time series) without manual preprocessing [9]. Moreover, some studies extend beyond

modulation classification to include joint tasks such as simultaneous SNR estimation and modulation format recognition using multilayer perceptrons (MLPs).

Authors surveyed the application of DL in wireless technology identification, which aims to classify communication standards rather than modulation types. This application involves electromagnetic environments characterized by heterogeneous and coexisting signals [9]. Recent work has demonstrated the feasibility of using DL architectures for tasks such as base station detection, interference signal recognition, and device identification.

The authors also review the challenges of feature engineering in traditional systems, emphasizing that DL eliminates the need for manually selected features, which are often unreliable or infeasible to estimate under practical constraints. They argue that the abundance of wireless data and recent progress in DL theory make these models highly suitable for scalable and adaptive WSR systems.

In other research, authors in [9] utilized AI-based load prediction to enhance the load balancing in the cellular network. Simulation results (see Figure 2-5. Simulation results of load balancing algorithms [9]) show that the linear ensemble model performed better than the traditional algorithms in predicting the resource status.

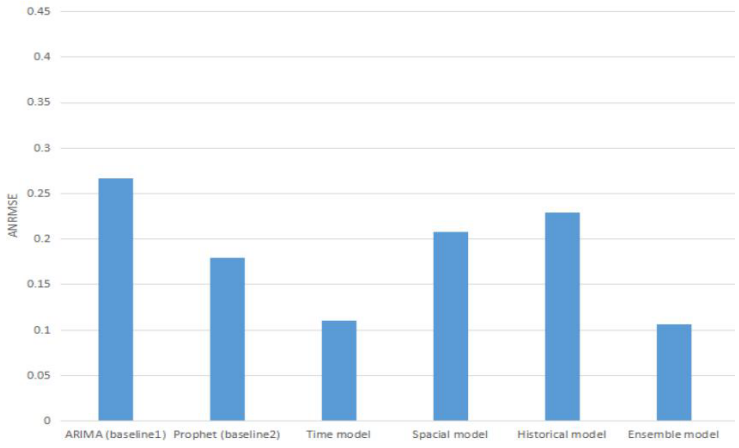


Figure 2-5. Simulation results of load balancing algorithms [9]

In [10] generative adversarial networks (GANs) are examined for channel modeling and estimation. NN is trained to capture the stochastic features of the channel that are difficult to be modeled in a closed analytical form [10]. Additionally, GANs can be trained to jointly optimize the

system performance metric simultaneously (bit or symbol error rate for example). Simulation results in [10] confirmed that GANs can converge to a low loss rate in terms of channel loss and symbol error rate, see Figure 2-6.

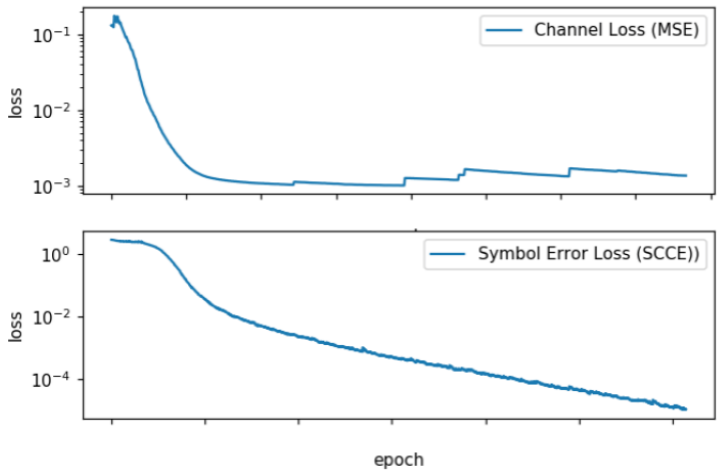


Figure 2-6. Loss of GANs Channel Estimation [10]

In [11] researchers replaced the complex signal detectors with DL-NN and computer vision algorithms for source detection and modulation classification. Gradient-weighted Class Activation Mapping (Grad-CAM) examines signal spectrogram collected in a surrounding environment and processes them through convolutional layers. At its output, it produces a localization map emphasizing the regions that form a unique signature of input. Grad-CAM output is fed to a CNN to obtain the class. The confusion matrix shown in Figure 2-7 illustrates the proposed solution and its performance as an RF classifier.

In [12], the authors explored the use of Reinforcement Learning (RL) as a promising approach for addressing energy optimization and resource allocation challenges in wireless networks. As the complexity and heterogeneity of wireless environments grow with the advancement of 5G and beyond, traditional optimization methods increasingly fall short in adaptability, scalability, and real-time responsiveness [12]. RL emerges as a suitable alternative by combining deep learning's representation capabilities with reinforcement learning’s decision-making framework.

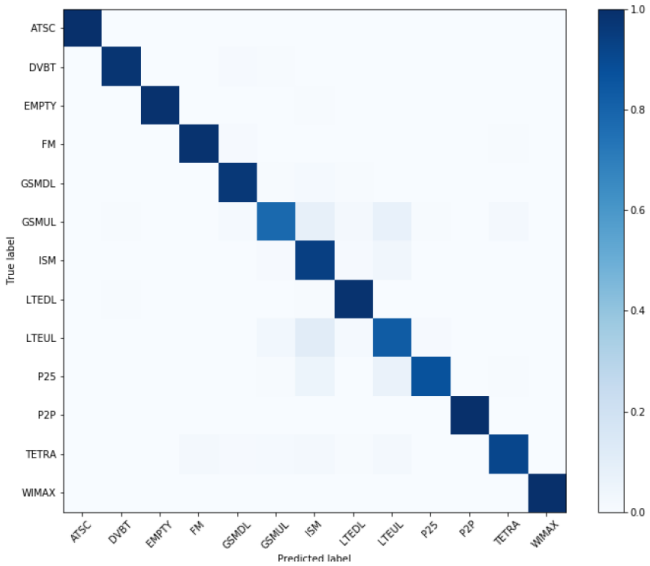


Figure 2-7. Confusion Matrix of RF Classifier proposed in [11]

Researchers highlight RL agents capability to operate efficiently in high-dimensional state spaces—typical of modern wireless systems. In addition, they confirmed that RL is uniquely positioned to handle the dynamic and stochastic nature of wireless networks, without requiring explicit models for the environment [12]. Authors identified relevant RL techniques, including Deep Q-Networks (DQNs), policy gradient methods, and actor-critic variants like A3C and DPPO, many of which have been successfully applied in other domains such as robotics and autonomous systems [12]. However, they also mentioned the current limitations, including the need for real-world testing, the exploration of diverse network scenarios, and the incorporation of additional network parameters such as user mobility and traffic heterogeneity.

So far, the related studies we have examined, have demonstrated DL efficiency in tackling sub-function-design(s) in the wireless communication system. Despite the performance optimization achieved in individual system design blocks reported by simulation results, it is still considered sub-optimal for the whole design. DL approach is capable of further enhancing the performance of wireless communication systems by jointly optimizing end-to-end functionality.

In [13] researchers proposed a novel approach that trains DL-NNs as an autoencoder (AE) to mimic the function of the whole wireless communication system including all its design blocks. AE uses unsupervised learning to compress the input data (at the encoder NN) to its efficient latent space [14]. Later, the compressed representation is fed to the decoder NN to reconstruct

the original input data. Thus, authors in [13] considered implementing the wireless transmitter, channel, and receiver as an NN-AE. The learning process of such wireless AE aims to optimize the system performance for a determined loss function. Unlike traditional algorithms for channel modeling, wireless AE can capture the non-linear patterns in system operations [13]. As a result, it can overcome the imperfections caused by the approximations in the conventional models. Moreover, utilizing parallel processing architectures can execute the learning process faster and with better energy efficient than sequential processing in traditional systems [15]. Thus, wireless AE offers end-to-end optimization at a lower cost compared to attempts aimed to jointly optimize multiple design blocks for conventional wireless systems [13]. Figure 2-8 shows the proposed structure of wireless AE proposed in [13]. Both transmitter and receiver consist of multiple dense feedforward layers. The encoder (transmitter) uses stochastic gradient descent (SGD) to learn how to find the latent representation $\mathbf{x}$ of the input vector $\boldsymbol{s} \in \mathcal{M}$, where $\mathcal{M}$ is the possible messages. At the other end, the decoder (receiver) learns to reconstruct the original input from the compressed latent variables. The layer with softmax activation function produces probability vector $\boldsymbol{p} \in (0,1)^{\mathcal{M}}$ that determines the probability of received symbol to match one of the possible transmissions.

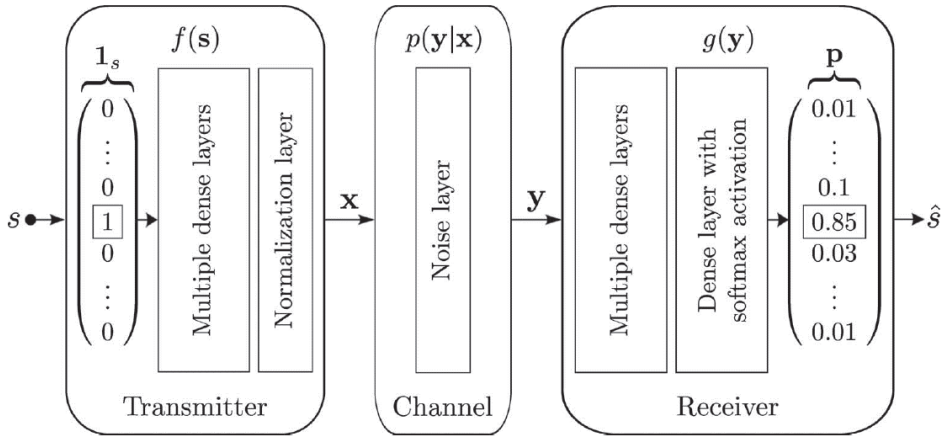


Figure 2-8. Wireless autoencoder over AWGN channel proposed in [13]

Simulation results in [12] for binary phase-shift keying (BPSK) modulation, show that the proposed wireless AE could outperform, in terms of block error rate (BLER), the conventional unencoded BPSK even for negative values of E_b/N_0 (see Figure 2-9).

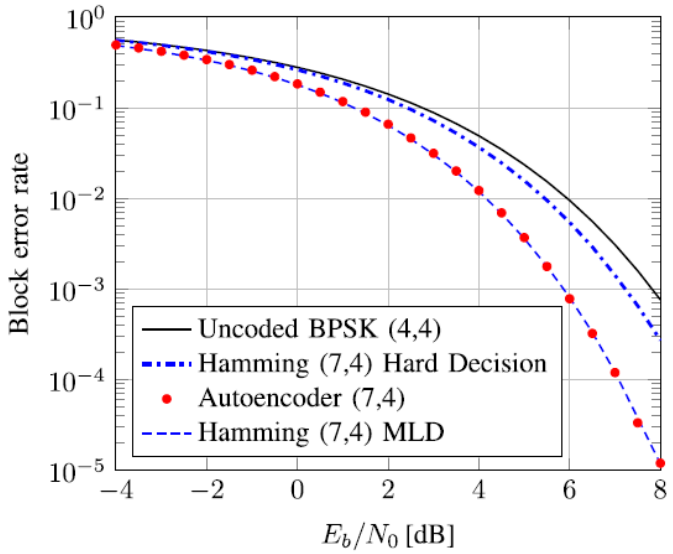


Figure 2-9. Wireless AE performance in BPSK [13]

However, without any prior information on the channel, the AE was able to learn the performance of BPSK with a maximum likelihood decoding (MLD) of Hamming (7,4). Comparable results were observed in the case of multiple users over an AWGN channel. Figure 2-10describes a wireless communication system with two transmitters and receivers. In this scenario, signals $x_1$and x_2 are interfering (mixed) at the receivers' terminals resulting in the received signals $y_1$and y_2 given in Eq. (1) and (2). It is observed that the authors neglect the path variations in the simulation.

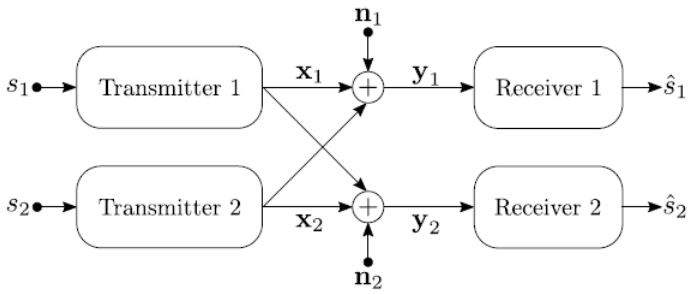


Figure 2-10. Multiple users wireless communication system [13]

$$y_1 = x_1 + x_2 + \mathbf{n}_1 \quad \text{Eq. (2.1)}$$

$$y_1 = x_2 + x_1 + \mathbf{n}_2 \quad \text{Eq. (2.2)}$$

Where $\mathbf{n}_1, \mathbf{n}_2 \sim \mathcal{CN}(0, \beta \boldsymbol{I}_n)$ are gaussian noise components [13]. Using the same AE structure for the single user introduced in Figure 2-11, BLER metric results for $(n, k) = \{(4,4), (4,8)\}$,

where n is the number of channel used, and k is the number of bits in the transmitted symbol [13] (see Figure 2-11). However, AE performance for $(n,k) = \{(1,1),(2,2)\}$ was identical to the conventional system [13].

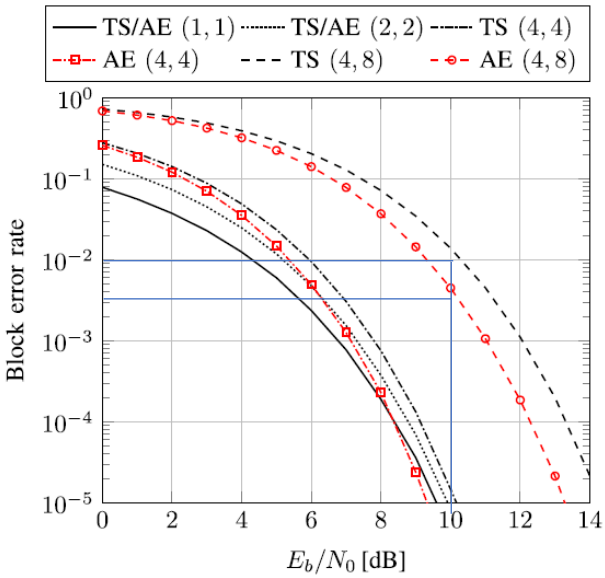


Figure 2-11. BLER performance of multiple users wireless AE [13]

Researchers in [15] extended the AE-based wireless communication to deploy multiple input – multiple output (MIMO) orthogonal frequency division multiplexing (OFDM) transmitter and receiver. Besides channel imperfections, they tackled the challenges of "limited training data" and external interference [15]. Data augmentation using a generative adversarial network (GAN) was proposed to generate "synthetic samples" and overcome the limitation of the training dataset [16]. Figure 2-12 shows the block diagram of the proposed system. Randomized smoothing and interference training are used to train the AE to suppress the external interference consequences [15].

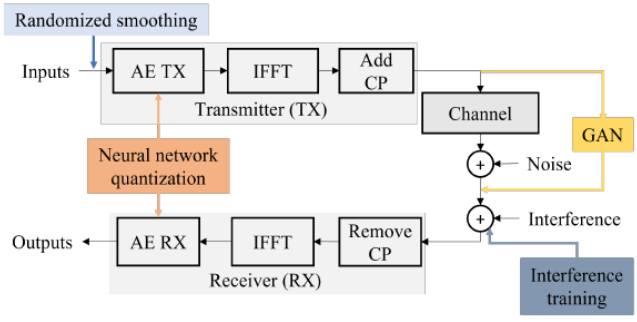


Figure 2-12. Block diagram of AE-OFDM systems with GAN and Interference suppression [15]

Bit error rate (BER) results of AE-OFDM system, in the absence of interference, confirmed the superiority of the AE-based approach over the conventional realization of wireless systems observed in [13] (see Figure 2-13).

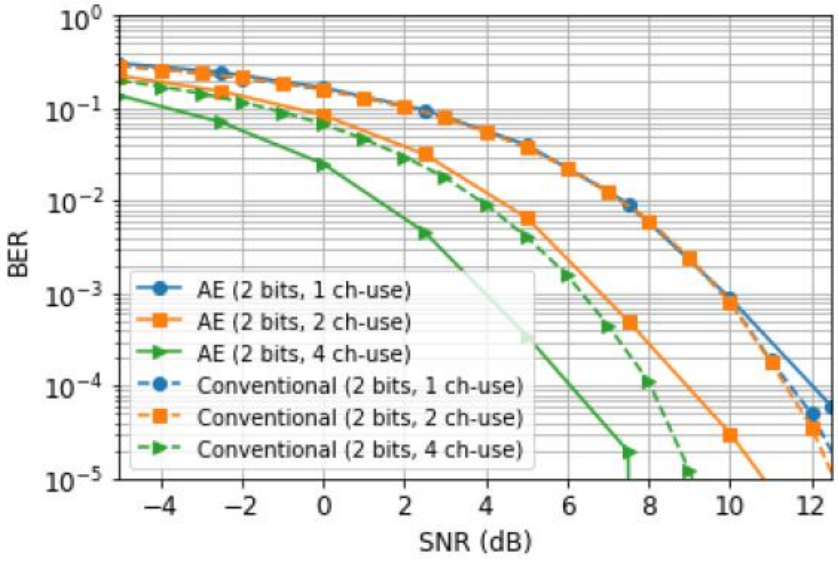


Figure 2-13. BER results of AE-OFDM system without interference [15]

Additional gain was observed in AE performance when additional impairments, like phase and frequency offsets, disturbed the channel (see Figure 2-14). This shows that AE was able to adjust its performance to enhance robustness against the additional distortions [15].

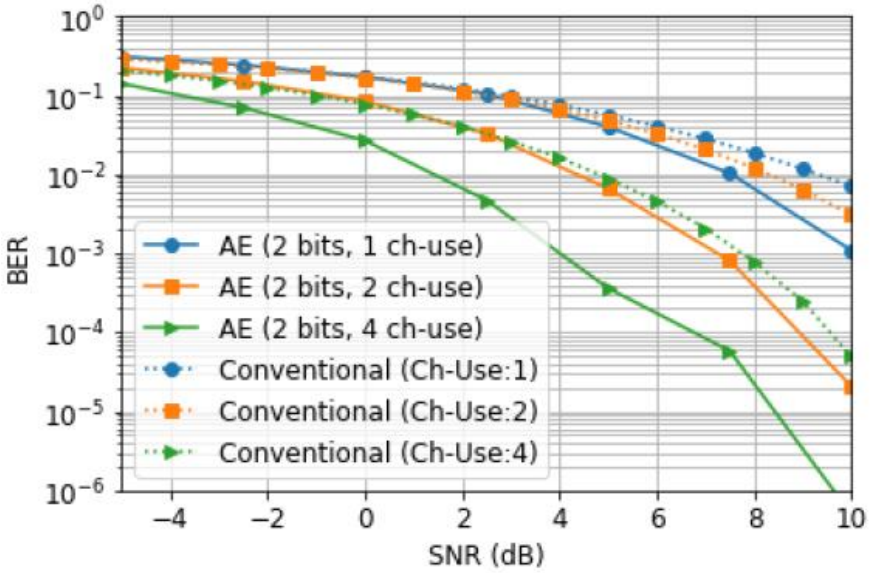


Figure 2-14. BER results of AE-OFDM with 10 degrees phase offset and 30 Hz frequency offset [15]

Further analysis demonstrates the AE-based approach capability of adaptively adjusting the constellation points to optimize the objective function (minimizing BER in this case) [15]. Figure 2-15 and Figure 2-16 illustrate the adaptive constellation generated by AE versus the rigid constellation for quadrature phase shift keying (QPSK) at 30 dB and -5dB SNR, respectively. It is obvious that AE precisely considers channel conditions in constellation forming.

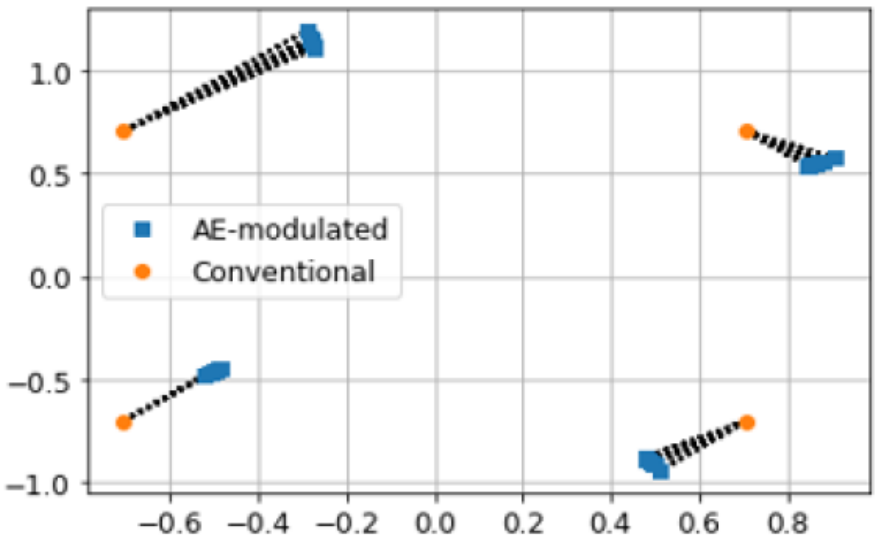


Figure 2-15. AE versus conventional constellations at 30dB SNR [15]

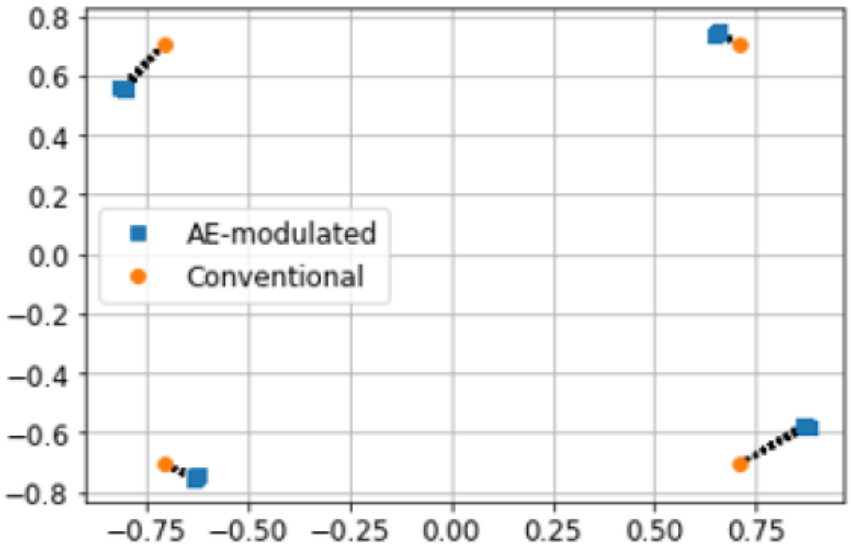


Figure 2-16. AE versus conventional constellations at -5dB SNR [15]

Lastly, researchers in [16] evaluated the gain in BER resulting from data augmentation using GAN. It was observed that data augmentation has a higher impact on smaller datasets more

willingly than larger datasets. Figure 2-17 and Figure 2-18 show the augmentation gain for 100 and 1000 sample original datasets, respectively.

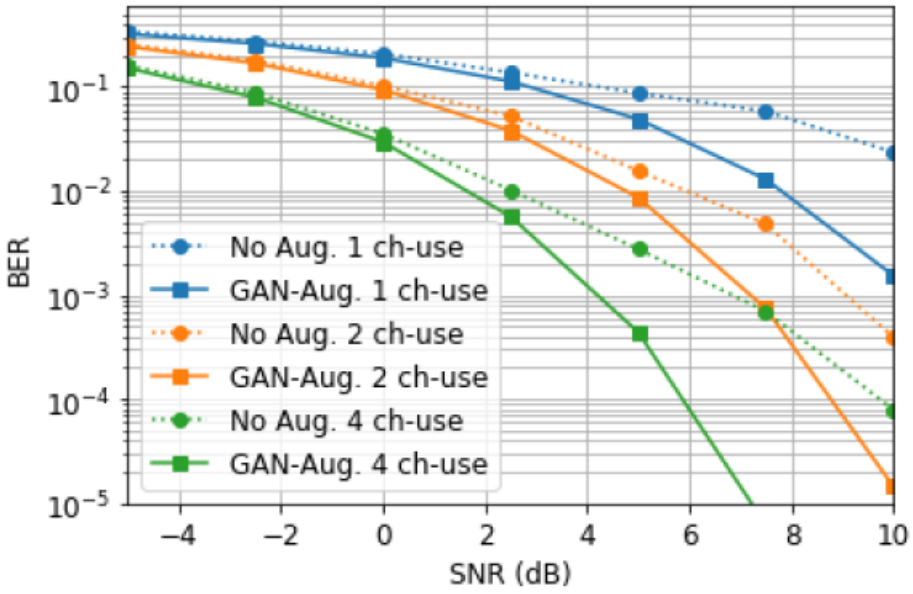


Figure 2-17. Data augmentation effect on AE performance (100 samples original dataset) [16]

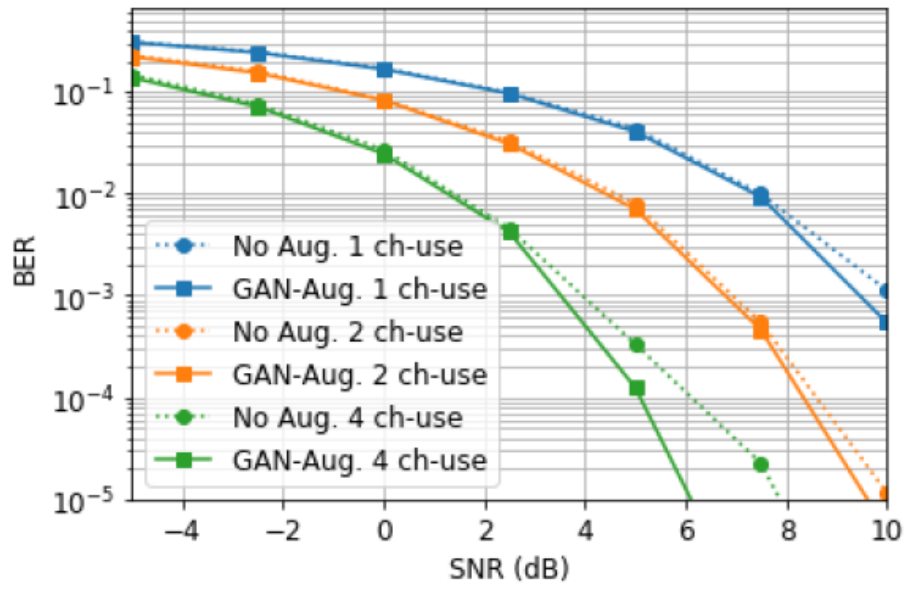


Figure 2-18. Data augmentation effect on AE performance (1000 samples original dataset) [16]

2.3 Challenges in DL-based wireless communication

DL has significantly contributed to performance enhancement in both block-design or overall wireless system design; however, several challenges still need to be addressed. Some of the challenges are:

- Number of possible messages: most of the simulations considered a limited number of possible messages to be exchanged between the transmitter and receiver. Conversely, practical scenarios include limitless possible messages [17]. There is an urge to investigate novel representations to exhaust potential messages.
- Training efficiency: it is obvious that performance optimization requires significant training to achieve a good DL-NN design. It is essential to consider the added overhead due to the training process. Furthermore, time-varying channels requires continuous recursive training whenever the NN design exceeds the tolerable deviation from the actual model [4]. A balanced trade-off between performance gain and computational and energy cost is substantially required.
- Full duplex channels: The literature still lacks studies on the DL interventions for full duplex channels. It is quite challenging to train the same NN to perform transmitting and receiving functions.
- Evaluation metric: Earlier research in this domain uses error-based metrics to evaluate the improvement in the system performance [17]. It is crucial to consider other factors while quantifying the performance. Time and power-based parameters should be considered while training process should consider jointly improving several aspects of the system.

In conclusion, data-driven approaches have been proven to produce more efficient solutions at lower levels of complexity. Research in DL-based optimization for wireless communication systems followed two different approaches in integrating DL solutions in the physical layer [13]. The first approach aims to improve the performance of sub-functions within a wireless system, including coding, signal detection, and modulation classification. The second approach targets the entire system jointly aiming to achieve end-to-end optimization. Wireless channel AE was proposed to accomplish this task. Leveraging the power of deep NN structures; AE was able to

outperform conventional wireless systems in different metrics (i.e., BLER and BER). Moreover, AE demonstrates high adaptivity to respond timely to variations in the wireless channel conditions. To overcome the small set of training data, data augmentation using GANs was able to enhance the performance. Overall, data-driven optimization has the following advantages over the traditional approaches [13]:

(1) Overcomes the channel estimation step by offering level system optimization.
(2) End-to-end optimization through system-level objective functions.
(3) Considering all data samples, rather than using only pilots, for training the DL model. This allows the investigation of vast optimization space during the training phase.
(4) Efficient training allows the NN to produce optimized solutions for new instances.

Yet, with all the improvements introduced by DL techniques in wireless communication systems, many challenges limit the practical implementation of these approaches. These challenges are great opportunities for research advancement in wireless communication based on DL approaches.

3 Autoencoder Principles, Types, and Applications

3.1 Introduction

Autoencoder (AE) is a deep learning (DL) neural network (NN) that comprises of two main parts: the encode and decoder. Using unsupervised learning, AE learns to compress the input data to its essential features at the encoder output. At the other end, the decoder learns to reconstruct the original data from the compressed representation [14]. AE uses unsupervised learning to capture the hidden patterns and features, called latent variables, which represent the data distributions. The encoder is trained to discover the sufficient latent variables that can be used by the decoder to precisely reconstruct the original data. Therefore, latent space encompasses only the most significant features of the input data [18].

3.2 Autoencoder Vs Conventional Encoder-Decoder

Compared to the typical encoder-decoder framework, AE's unique approach makes it suitable for unconventional tasks, like generative tasks in image processing and time series domain [14]. Most conventional decoders produce outputs that are different from the original input of the system. For example, in U-Net used for image segmentation, the encoder analyzes the input image to extract features and find the semantic classification of pixels. Accordingly, the decoder utilizes the feature map and pixel classes to construct objects or region segmentation masks [19]. In addition, the traditional framework utilizes a human-labeled “ground truth” to optimize the objective function. Hence, it uses a supervised learning strategy. In contrast, AE decoders use unlabeled data (unsupervised learning) to recover the data fed to the encoder [18]. Unlike the fully unsupervised learning strategy, AE uses the input data as the “desired output” to evaluate the training progression. Therefore, it is called “self-supervised” learning [20]. Generally, AEs have attractive attributes that make them an effective solution for several problems.

- Self-supervised learning strategy combines the advantages of reducing the reliance on human expertise and benefiting from the use of a ground truth to evaluate the learning effectiveness.
- In addition to the traditional DL applications, like dimensionality reduction and denoising, autoencoders can be used for several tasks related to information theory, component analysis, statistical inference, and wireless communications.
- Autoencoders can be composed in modular structures. Modular units can be interconnected in different ways to perform diverse tasks.
- Unlike principal component analysis (PCA), the autoencoder is capable of capturing complex nonlinear correlations.

3.3 Autoencoder Structure and Work Principles

Figure 3-1 shows the general structure of the autoencoder, consisting of three functional parts:

1. Encoder: multilayers with progressively a smaller number of nodes [14]. Encoder layers process (encode) the input samples to extract the compressed representation.

2. Bottleneck (latent space): denotes the most compressed form of the data.
3. Decoder: processes the latent variable through progressively larger layers to reconstruct the original input data.

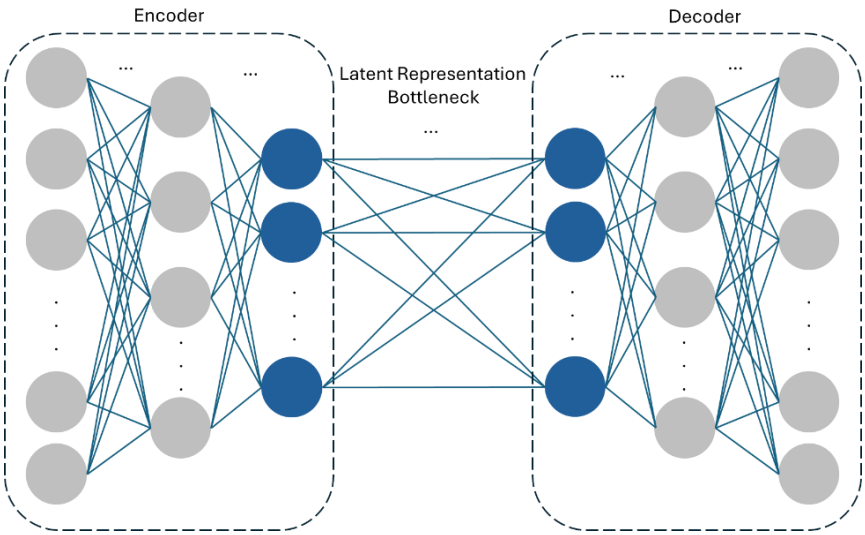


Figure 3-1. Autoencoder structure

In general, encoder-decoder is denoted as $E(n, m, p)$, where:

- n : number of nodes in the input layer.
- m: number of nodes in the hidden layers.
- p: number of nodes in the output layer.

However, from the definition of autoencoder $n = p$, and we will denote the autoencoder as $AE(n, m, n)$, Figure 3-2 illustrates the structure for modeling purposes. To obtain a mathematical model of an autoencoder, let us define the following [18]:

- $\mathbb{M}^n$: input data space
- $\mathbb{L}^m$: latent variables space
- $\mathcal{C}$: class of functions that encode data from $\mathbb{M}^n$to $\mathbb{L}^m$
- $\mathcal{D}$: class of functions that decode latent variables from $\mathbb{L}^m$ to $\mathbb{M}^n$
- $\chi = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k\}$ set of training vectors.

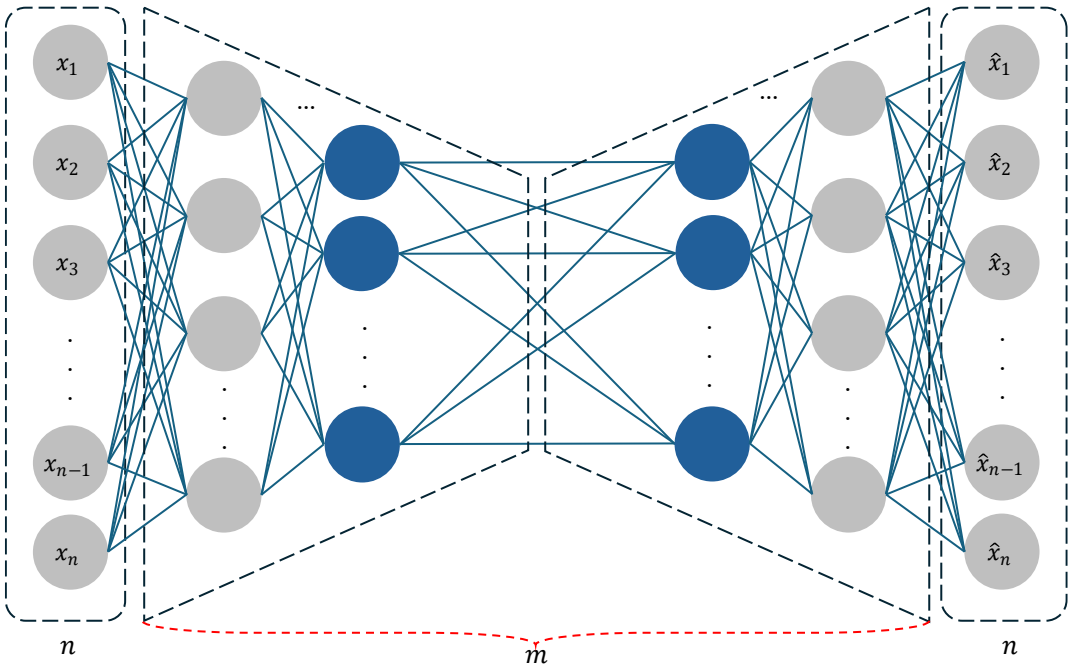


Figure 3-2. Autoencoder layers

Then we can formulate the autoencoder as an optimization problem to identify both functions $C \in \mathcal{C}$ and $D \in \mathcal{D}$ that minimize the error.

$$\min_{\mathrm{C,D}} \mathcal{E}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \min_{\mathrm{C,D}} \sum_{k=1}^{K} \mathcal{E}(x_k, \hat{x}_k) \qquad \text{Eq. (3.1)}$$

We can express the output of autoencoder $\hat{x}_k$ as:

$$\hat{x}_k = D.C(x_k) \qquad \text{Eq. (3.2)}$$

Then Eq. (3.3) can be written as:

$$\min_{\mathrm{C,D}} \mathcal{E}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \min_{\mathrm{C,D}} \sum_{k=1}^{K} \mathcal{E}(x_k, D \circ C(x_k)) \qquad \text{Eq. (3.3)}$$

Error function should be selected carefully to reflect accurately the performance of the autoencoder [21].

From an information theory perspective, basic autoencoder is approximated as feedforward NN [21]. Thus, input signal propagates from input to output, while error reversely propagates from output layer to the input layer (Backpropagation). At any point, state depends only on one previous step, hence, follows Markov chain [21]. Therefore, the data processing inequality (DPI) of feedforward NNs applies for both propagation directions. In forward path (2.4) shows the DPI for the data propagation input layer to the output layer.

$$\mathbf{I}(\mathrm{X};\mathrm{h}_1) \geq \mathbf{I}(\mathrm{X};\mathrm{h}_2) \geq \cdots \mathbf{I}(\mathrm{X};h_L) \geq \mathbf{I}(\mathrm{X};Y) \quad \text{Eq. (3.4)}$$

where X is a random variable that represents the input, $\mathrm{h}_1, h_2, \ldots, h_L$ are the hidden layers, and Y is the output random variable. (2.5) gives the DPI of backward propagating error.

$$\mathbf{I}(\delta_\mathrm{L};\delta_{L-1}) \geq \mathbf{I}(\delta_\mathrm{L};\delta_{L-2}) \geq \cdots \mathbf{I}(\delta_\mathrm{L};\delta_1) \quad \text{Eq. (3.5)}$$

Considering a symmetric autoencoder, we will use s_i to indicate to the state at each hidden layer of the encoder portion. For the decoder, state at each hidden layer will be indicated with s_i'. Finally, latent space will be represented by the random variable Z(see Figure 3-3).

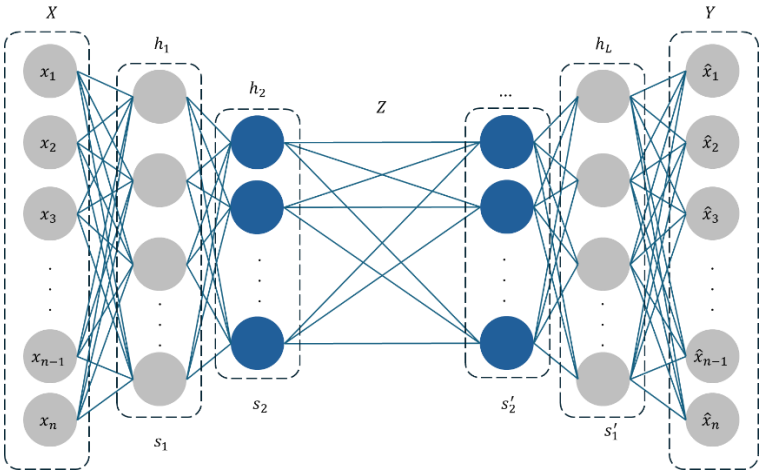


Figure 3-3. Autoencoder hidden layers and states

Then we can obtain the probabilistic form of the encoder and decoder parts using chain rule, with respect to the latent space, as given in Eq. (3.6) and Eq. (3.7) respectively.

$$P(Z|X) = \int P(s_1|X)\, P(s_2|s_1) \ldots P\left(Z\middle|s_{\frac{m}{2}}\right) dt_1\, dt_2\, dt_{\frac{m}{2}} \quad \text{Eq. (3.6)}$$

$$P(Y|Z) = \int P(Y|s_1')\, P(s_1'|s_2') \ldots P\left(s_{\frac{m}{2}}'\middle|Z\right) dt_1'\, dt_2'\, dt_{\frac{m}{2}}' \quad \text{Eq. (3.7)}$$

Equations (2.6) and (2.7) could be represented using a graph as in Figure 3-4.

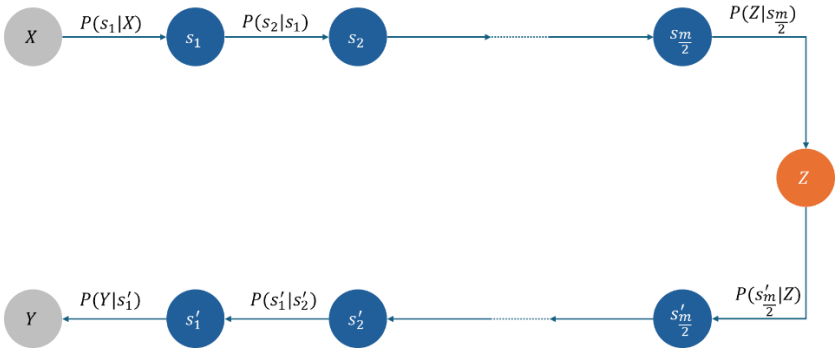


Figure 3-4. Graph representation of a probabilistic model of autoencoder [21] (Modified)

3.4 Autoencoder Hyperparameters

Hyperparameters are essential for autoencoder learning process. Improper selection of their values could lead to a nonoptimal performance or even cause the autoencoder to fail. Most important hyperparameters of autoencoder are:

1. Autoencoder depth represents the number of hidden layers between the input and the output. Autoencoder depth affects the complexity of the model, learning, and processing times.
2. Layer size determines the number of nodes in each layer. Input and output layer sizes vary according to the data type. Images require quite a large number of nodes [22]. However, in basic autoencoder hidden layer sizes decrease progressively at the encoder to reach the required size for the latent space. The opposite happens in the decoder to reconstruct the original input [14].
3. Latent space size defines the compression ratio of the autoencoder [22]. It is crucial to select the proper size of latent space. Extremely large latent space might result in overfitting. Contrarywise, extra small latent space is more likely to cause autoencoder to underfit the data [23].
4. Loss function evaluates the efficiency of autoencoder learning; appropriate selection loss function gauges the reconstruction loss. Furthermore, loss function plays a fundamental role in model optimization [14].
5. Activation functions could be used to obtain the neuron output. Nonlinearity, output range, computational cost, sparsity, and many other factors influence the selection of activation functions [24].

Nonetheless, different variations of autoencoder have more hyperparameters. For instance, dense autoencoders might use dropout to discard a fraction of nodes to avoid overfitting [22]. Hence, the dropout rate should be tuned carefully to attain the optimal performance.

3.5 Autoencoder Types and Applications:

Evolving applications of autoencoders require structural variations to adapt to the required performance.

Undercomplete autoencoders are simple structures aimed to perform dimensionality reduction tasks. Using the bottleneck, which is smaller than the input and output layers, produces a compressed form of the input data [14]. Using suitable activation functions, undercomplete autoencoders can capture nonlinear correlations, called manifold learning [23]. Hence, it

outperforms the PCA that can capture only linear relationships (see Figure 3-5). Their fast and accurate performance make them suitable for data preprocessing [25]. In addition, they are capable of large-scale non-linear applications [23]. However, undercomplete autoencoders are prone to overfit the training data through learning "identity function". More precisely, inadequately limited latent space allows for minimizing the loss by replicating the input data at the output without sufficient intermediate compression [25].

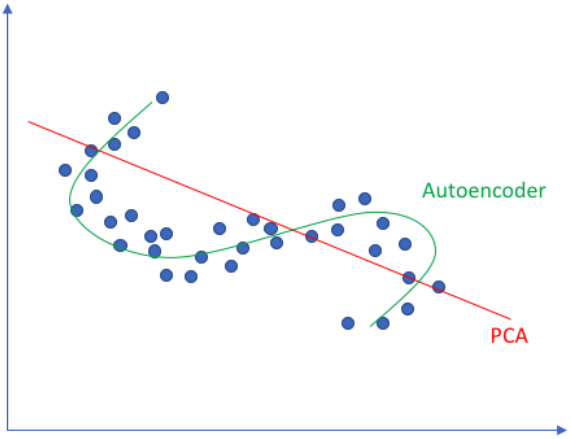


Figure 3-5. PCA versus Autoencoder Dimensionality Reduction [27]

To address the overfitting drawback of undercomplete autoencoder, regularized autoencoders introduce a modified method of loss metric. Sparse autoencoders establish bottleneck by varying the number of nodes to be activated simultaneously (see Figure 3-6) [26]. Unlike undercomplete autoencoders that operate the entire NN at each instance, sparse autoencoders are "penalized" to keep the number of activated nodes below a certain threshold. Hence, overfitting risk is mitigated without compromising the network capacity. Reconstruction error is regularized by sparsity function to impose learning the efficient latent space under sparsity constraints [25]. Kullback-Leibler (KL) divergence is used to quantify the deviation between the learned distribution of node activations and the desired one [14].

Contractive autoencoders are immune to noisy input data. Therefore, they can, effectively, capture the important hidden patterns of the data. Jacobian matrix gauge network gradients respond to changes in the input data. Accordingly, autoencoder is penalized for altering the output in response to insignificant variation in the input signal.

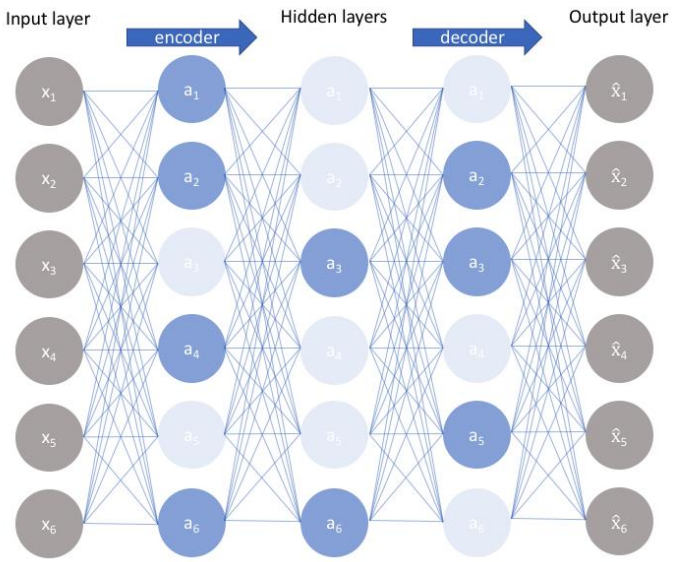


Figure 3-6. Sparse Autoencoder [27]

This penalty term limits the loss function from observing the minor noise in the input data. As a result, latent variables represent only the most substantial features [25]. Figure 3-7 shows the principle of contractive autoencoders. The noisy input values in the vicinity of training observations are contracted to produce a constant output.

As formentioned, the types we discussed are trained to reconstruct their own input. However, denoising autoencoders are exposed to corrupted input data and learn to remove unwanted noise. To accomplish this goal, autoencoder is fed with clean data artificially disturbed by additive Gaussian noise [26]. Learning process trains the autoencoder to filter out the noise by minimizing the reconstruction error against the clean data. In image processing applications, traditional denoising techniques require prior knowledge of noise type and parameters to reduce the noise effectively. Image-denoising autoencoders can learn to extract the image regardless of the noise nature.

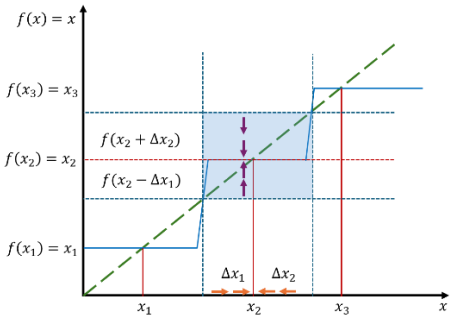


Figure 3-7. Working Principle of Contractive Autoencoders [27]

As mentioned earlier in this chapter, autoencoders can effectively serve as generative models. Variational autoencoders (VAEs) learn the probability distributions of the compressed representations. Trained VAEs generate new data points that have distributions that are variations of learned ones [14]. VAEs surpass the other types of autoencoders by learning a continuous latent space that represents the means and standard deviations of the learned compressed distributions. Hence, VAEs are considered as "stochastic" encoders [14]. Loss function is regularized by KL-divergence between the prior distribution (input data) and posterior distribution (latent variables).

3.6 Autoencoder for wireless communication

We presented in Chapter 2 a variety of related studies on optimizing wireless communications performance using deep learning techniques. Research [1] through [12] discussed the optimization of subfunctions in wireless communication systems. On the other hand, studies in [13] through [27] ,[28], and [29] presented a holistic framework for end-to-end optimization using the concept of autoencoder. Following, we will discuss the various methods of implementing wireless autoencoder.

Wireless channel autoencoders can be classified into model-assumed autoencoder and model-free autoencoder [30]. In model-assumed autoencoder channel impairment parameters are considered during the construction of the autoencoder. These parameters include but are not limited to noise profile, fading type, frequency and phase offsets, and delay [30]. To compensate for these imperfections, radio transformer network (RTN) is integrated with the decoding NN of the autoencoder. RTN comprises a parameter estimation network that learns to estimate channel parameters from the received signal. Estimated parameters are fed to a transformation layer that compensates for channel distortions [30]. Figure 3-8 illustrates an autoencoder receiver with RTN.

To train the RTN to effectively inverse the channel impact, the channel model and parameters should be precisely characterized. Thus, the receiver is trained using a pre-assumed channel model and then fine-tuned using the actual channel [17].

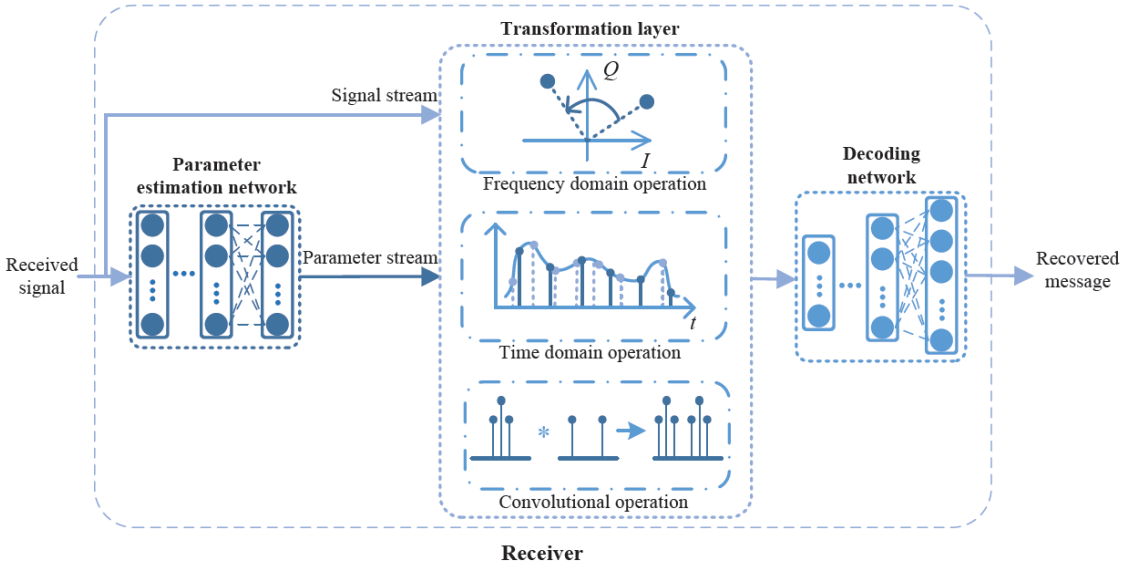


Figure 3-8. Autoencoder Receiver with RTN [17]

However, RTN-based receiver lacks transmitter gradients. Therefore, researchers in [31-33] proposed utilizing generative adversarial networks (GAN) to link transmitter and receiver networks [31]. Figure 3-9 shows the structure of the GAN-based autoencoder. Efficiently trained GAN can model the channel accurately and backpropagate the transmitter gradients [32][33][34].

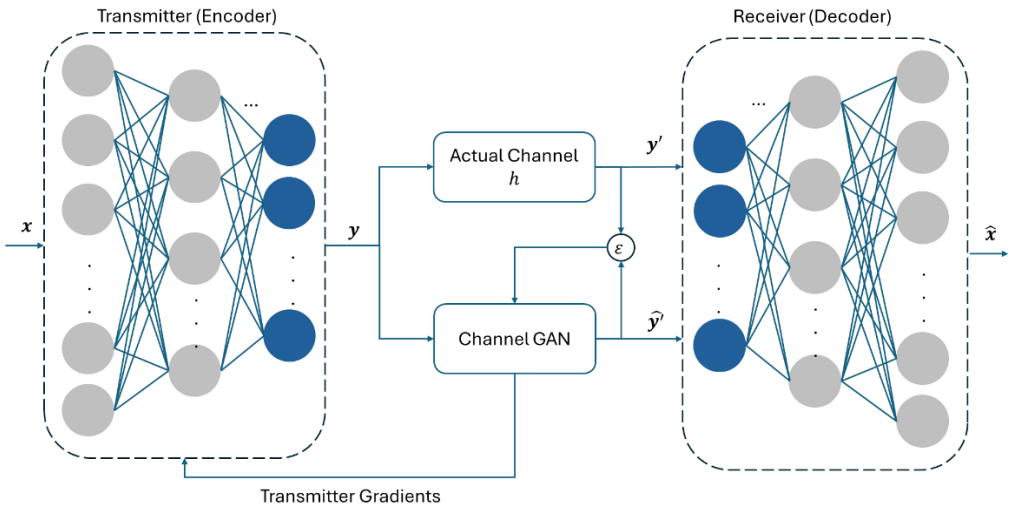


Figure 3-9. GAN-based Wireless Autoencoder [31]

In summary, autoencoder is an efficient self-supervised learning technique for data compression and reconstruction. Different variations of autoencoder showed a competitive performance over a wide range of applications. In wireless communications, autoencoder has proven a significant potential for achieving end-to-end optimization. Its ability to learn channel

impairments analyzing the over-air signal overrides a challenging process of channel estimation and approximation. However, most of the presented related studies depicts only simulation-based performance. Hence, the concept of wireless autoencoder needs to be investigated using data collected under real-wrold communication scenarios.

4 Deep Learning Decoders in Optical Wireless Communication Operating under Atmospheric Challenges

4.1 Introduction

The increasing quest for high-speed, secure, and interference-robust communication has driven many advancements in Optical Wireless Communication (OWC) system1 designs. Owing to their inherent design characteristics, like high bandwidth and long communication ranges, OWC systems are widely used for indoor networks, vehicular communication, and military applications2,3. However, the performance of OWC systems is highly prone to atmospheric conditions, including fog, rain, wind, and temperature fluctuations, which introduce turbulence that degrades the signal quality [35][36].

Traditional OWC systems rely on classical signal processing techniques for encoding and decoding. Theses algorithms perform adequately under controlled conditions, but struggle with unpredictable atmospheric variations in dynamic environments [37]. Consequently, there is an increasing need for adaptive solutions to enhance system resilience against real-time channel impairments. Deep learning (DL) offers a promising approach, which can learn complex patterns and adapt to nonlinearties caused by atmospheric turbulence [38].

Most existing research on DL-based OWC systems rely heavily on simulations, and often assume simplified channel models like additive white gaussian noise (AWGN). While these models are useful for theoretical insights, they fail to capture real-world atmospheric variations8.

This study implements a DL-based decoder for an OWC system using empirical data collected in a controlled laboratory environment. The experimental setup features a weather chamber capable of precise fog injection, temperature control, and internal fans to emulate atmospheric turbulence. Data transmission is conducted using a 1310 nm wavelength with On-Off Keying (OOK) modulation, and the pair of transmitted and received data are used to train and evaluate the DL-based decoder.

This study makes several key contributions to advance OWC system designs:

1) DL-Based Decoder Implementation: Developed a deep learning-based decoder to mitigate atmospheric turbulence, offering a strong alternative to traditional methods.
2) Empirical Data Utilization: Utilized empirical data from a controlled weather chamber to evaluate OWC performance beyond AWGN-based simulations, which provides more accurate and realistic assessment.
3) Performance Assessment: Assessed system performance under varying environmental conditions, demonstrating significant improvements in the block error rate (BLER) in comparison to conventional methods.
4) Scalability and Practical Deployment: Provided insights into the practical deployment of DL-based decoders in dynamic OWC environments, highlighting their potential to enhance communication performance in real-world applications.

4.2 Related Work

In [39], simulation-generated data were used to train a fully connected deep learning neural network (DL-NN) to mimic the function of a model-based OWC system. Building on the RF autoencoder approach proposed in [40], the authors designed an encoder (transmitter) using multiple dense layers and a normalization layer to generate OOK waveforms. To simulate channel effects, an AWGN layer with constant variance was applied. At the receiver side, the decoder used multiple dense layers to reconstruct the original data. BLER simulations demonstrated that the proposed DL-based system outperformed conventional OWC systems8. However, the study did not provide insights into training efficiency, accuracy, and cost function. Moreover, the study relied on a time-invariant AWGN model, limiting real-world applicability.

Expanding on the end-to-end OWC autoencoder concept, the authors of [41] explored various modulation schemes and introduced a multipath fading model along with AWGN. The proposed approach employs multiple dense layers with rectified linear unit (ReLU) activation to function as an orthogonal frequency-division multiplexing (OFDM) transmitter [40]. However, key conventional design elements such as Inverse Fast Fourier Transform (IFFT), negative clipping, and cyclic prefix (CP) insertion were retained. The generated constellations were passed through AWGN and multipath fading channels to simulate atmospheric turbulence. At the receiver, traditional CP removal and FFT processing were followed by trained dense layers to decode and reconstruct the transmitted symbols. While DL-based system performed comparably to

conventional 16-QAM systems at lower $Eb/N0$ values (< 25 dB), it showed only slight gains for QPSK modulation within 15 dB to 23 dB range. For multipath fading conditions, the proposed system outperformed conventional methods across both QPSK and 16-QAM schemes [41].

In [42] and [43], DL-based transceivers were implemented for free-space optical multiple-input multiple-output (FSO-MIMO) systems to mitigate simulated atmospheric turbulence effects. Initially, the conventional detector was replaced with a DL-based detector comprising a dense layer, dropout layer, convolutional layer, and softmax layer [42], while retaining conventional channel estimation and demodulation methods. The DL-based receiver achieved a similar symbol error rate (SER) performance to existing ML-based receivers [43], but with a lower computational complexity of O(M/2) compared to O(M) for ML-based methods. However, the study did not compare DL-based performance with conventional FSO-MIMO systems.

In subsequent study, both the transmitter and receiver were implemented using DL-NNs, with some traditional processing blocks maintained [44]. The autoencoder approach demonstrated reduced complexity compared to ML-based systems while maintaining comparable SER performance.

Despite promising outcomes, all aforementioned studies relied on simulation-generated data and approximated mathematical models to characterize channel impairments. These implementations limit their applicability in real-world conditions.

In contrast, researchers in [45] investigated ML techniques for separating multiple data streams from optical transmitters under realistic conditions. Real-time experimental data were collected using a 1310 nm laser source and two independent pseudo-random sequence (PRBS) generators. A weather chamber was utilized to emulate atmospheric turbulence, incorporating temperature variations up to 58.61°C, wind speeds of 14.21 m/s, and relative humidity of 80.56%. The received signals were processed using ML algorithms, and the results showed above 90% validation accuracy for traditional classifiers such as SVM, k-NN, ANN, Ensemble, and Decision Trees [45].

While [45] leveraged real-world data, it primarily explored traditional ML approaches rather than end-to-end DL-based frameworks. This highlights the existing gap in leveraging deep

learning for OWC systems by using real experimental data. This work bridges the gap utilizing empirical data from controlled weather chamber, providing a more realistic assessment of the DL-based OWC decoder performance under diverse atmospheric conditions.

4.3 Experimental Setup

To evaluate the performance of optical wireless communication systems under realistic environmental conditions, a controlled experimental setup was developed to simulate atmospheric turbulence and collect transmission data. The primary goal of this setup was to generate labeled datasets representing the effect of fog, temperature variation, and beam misalignment on received optical waveforms, which were later used to train and test learning-based detection algorithms. All data was generated in a laboratory environment where atmospheric conditions could be precisely manipulated and repeated.

Figure 4-1 illustrates the experimental setup used for data collection under turbulence-induced impairments. The optical transceiver employed a small form-factor pluggable (SFP) module operating at a wavelength of 1310 nm with an output power of 4 dBm. The SFP was driven by a SIGLENT SDG6032X signal generator, which generated a pseudorandom binary sequence (PRBS) at a data rate of 140 Mbps with a 50% duty cycle and a peak-to-peak voltage of 2 V. The data stream was modulated using intensity modulation with direct detection (IM/DD) and transmitted through a Thorlabs F810APC-1310 collimator to maintain beam quality and alignment.

On the receiver side, a Thorlabs DET08CFC photodetector converted the received optical signal into an electrical waveform. This detector supports a wavelength range of 800–1700 nm and offers a 5 GHz bandwidth at -3 dB, ensuring high-fidelity signal capture. Environmental factors such as fog density, airflow, and temperature were varied systematically during data collection to reflect different levels of atmospheric distortion.

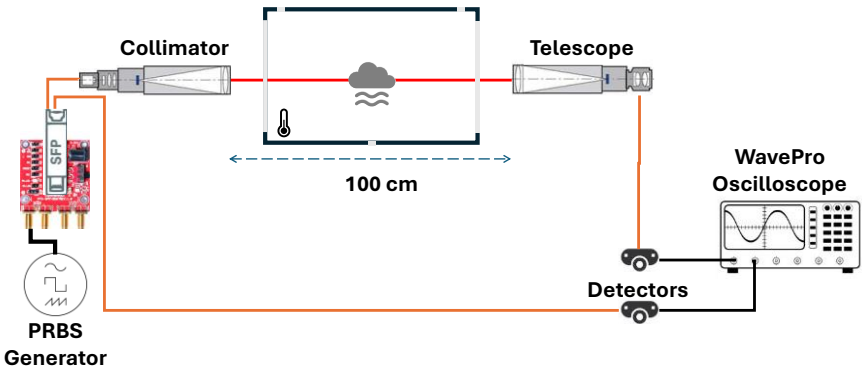


Figure 4-1 Experimental setup for weather turbulence of OWC link

To create atmospheric turbulence, a closed weather chamber was utilized, featuring heating coils to control temperature, four strategically positioned fans to create turbulence, and three fog injection pathways to introduce moisture (fog). The chamber enables the generation of dynamic environmental conditions to replicate real-world weather scenarios. Two fans were mounted at the bottom, while the remaining two fans were affixed to the sidewalls to ensure uniform airflow distribution. The wind speed was maintained at approximately 11.17 m/s (~25 mph), classified as "strong breeze" per the Beaufort wind scale. The environmental conditions within the chamber were controlled using two humidifiers along with integrated temperature and humidity sensors. The relative humidity can be adjusted between 40% and 90%, while the temperature is set to 70 °F, enabling comprehensive performance testing under varying fog densities. Figure 4-2 presents the actual lab setup for data collection.

Figure 4-2 Lab experimental setup for data collection

Data were recorded using a WavePro 254HD-MS oscilloscope at a sampling rate of 20 Gsamples/s. Each acquisition captured 5,000 samples and 10,000 acquisitions were collected for each atmospheric turbulence scenario. The data collection process involves capturing the optical signal intensity from an optical detector under varying atmospheric conditions. The detected optical signal was converted into an electrical waveform from which bit streams were extracted via thresholding at half the peak-to-peak voltage. The extracted bits were grouped into sets of 1,000 bits for subsequent processing. Figure 4-3 compares received data before and after the atmospheric turbulence, highlighting signal degradation.

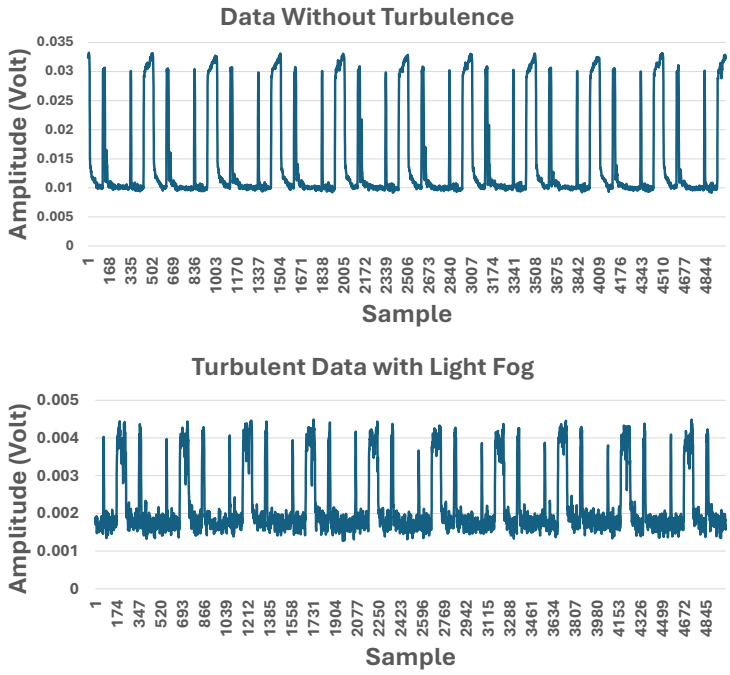


Figure 4-3 Received data waveforms.

To evaluate the performance of the system, holdout cross-validation was applied with a 70:30 split, in which 70% of the data were allocated for training and 30% for validation. This approach ensures a comprehensive evaluation of the system's ability to generalize under diverse turbulence conditions.

4.3.1 Channel Sounding

Channel sounding was conducted to establish a baseline for the optical channel and verify that the laboratory environment, in the absence of turbulence, is optically quiet [46]. This assessment confirmed that the channel exhibited minimal background noise and interference and remained

time-invariant under controlled conditions. Multiple pulses were transmitted to characterize the channel, and the received power was measured over propagation distances from 10 cm to 100 cm. The pulse duration was set to 1 ns, and the pulse repetition frequency (PRF) was 100 Hz to ensure accurate measurements of the channel response [47].

The measured power profile confirms that the channel adheres to the free-space path-loss model given in Eq.(4.1).

$$PL(d) = 20\,log_{10}(d) + 20\,log_{10}(c/\lambda\,) - 147.55 \qquad \text{Eq. (4.1)}$$

where d is the transmission distance in meters, and λ is the wavelength in meters. Results indicated a power loss of approximately 20 dB per decade of distance, consistent with theoretical expectations for free-space optical propagation.

Figure 4-4 shows the average of the transmitted pulses, with the corresponding average received pulses at 70 cm from the transmitter and the power profile.

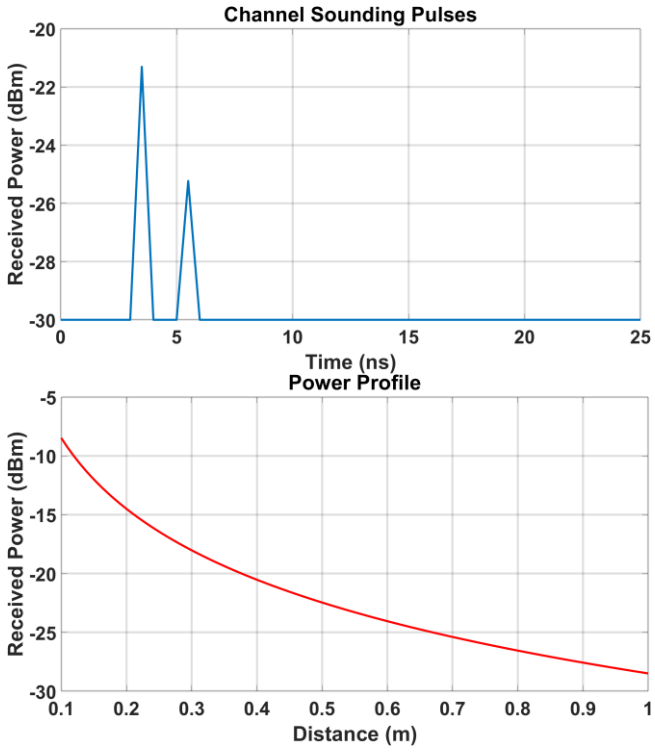


Figure 4-4 Sample of channel-sounding pulses and channel power profile

4.3.2 Atmospheric Turbulence

Atmospheric turbulence was induced by injecting fog at varying densities into the chamber and adjusting the internal temperature. Attenuation was measured as a function of fog density to quantify performance degradation caused by atmospheric turbulence. The fog density was varied following the ITU-R P.840-8 recommendation [48], ranging from light fog (less than 0.01 g/m³ of liquid water content (LWC)) to thick fog (~0.5 g/m³ LWC). Figure 4-5. illustrates the measured attenuation as a function of fog density at a fixed temperature of 70°F, and temperature under thick fog conditions (~0.5 g/m³ LWC).

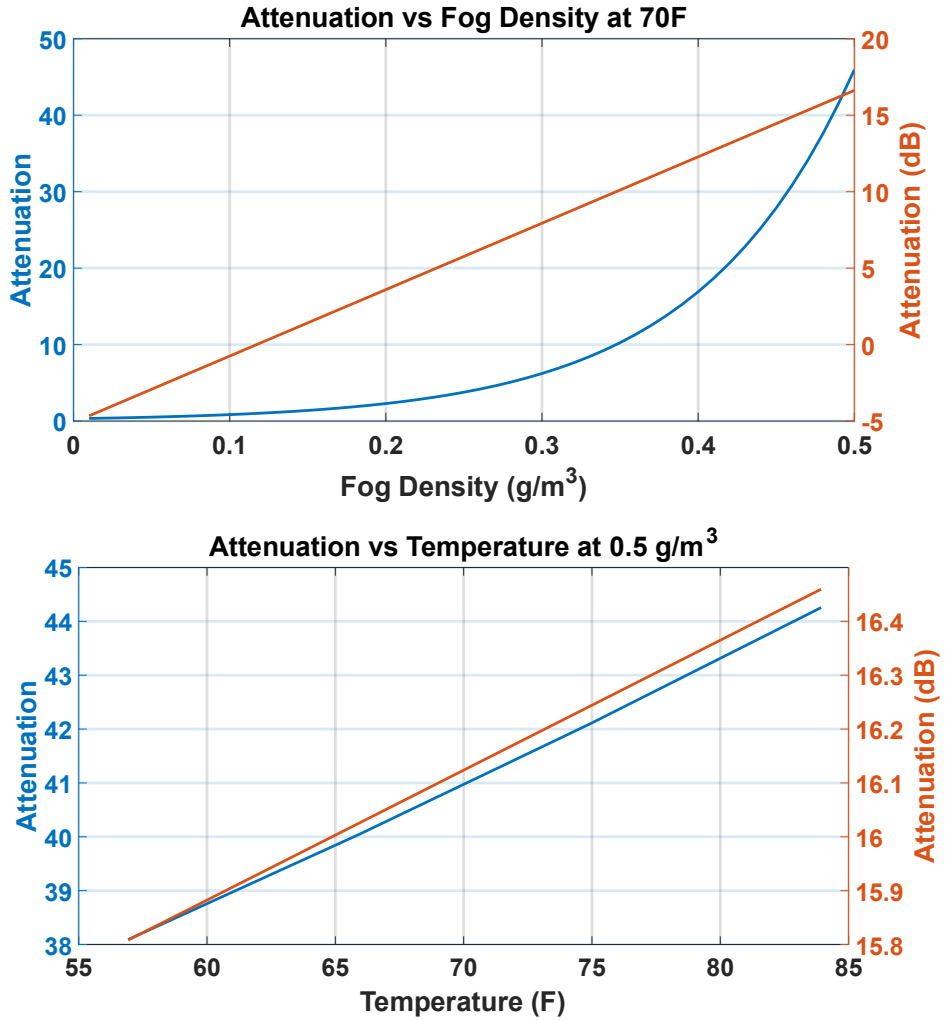


Figure 4-5 Channel attenuation as a function of fog density and temperature

Atmospheric turbulence significantly affects the transmitted optical signal, resulting in power loss, random fluctuations in the received signal intensity, and the overall degradation of the

communication link quality [49]. To further characterize the turbulent channel, eye diagrams were analyzed to assess signal integrity. Figure 4-6 depicts eye diagrams under light, moderate, and thick fog conditions. These visual representations provide insights into signal integrity degradation and increased intersymbol interference as turbulence intensifies.

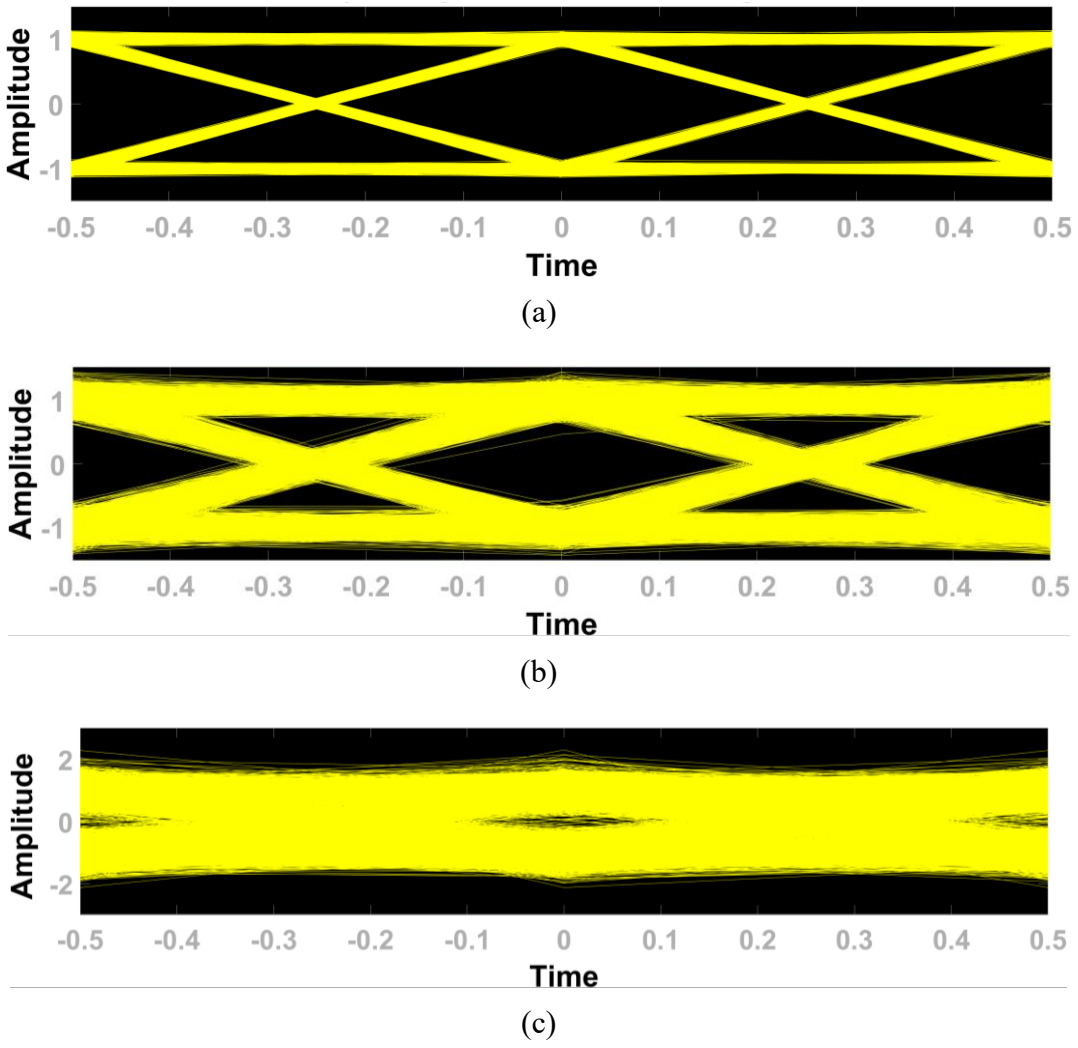


Figure 4-6 Eye diagram (a) under light fog, (b) under moderate fog, (c) under thick fog

To statistically characterize the turbulent channel, we applied the L-moments method, which estimates empirical distribution parameters using probability-weighted moments (PWM) as defined in Eq. (4.2).

$$\beta_r = \frac{1}{n}\sum_{k=r+1}^{n}\binom{k-1}{r}\binom{n-1}{r}^{-1} X_{(k)} \qquad \text{Eq. (4.2)}$$

where $X_{(k)}$ is the k_{th} data point and r is the moment order. L-moments of different orders can then be derived using Eq. (4.3), Eq. (4.4), Eq. (4.5), and Eq. (4.6).

$$\lambda_1 = \beta_0 \quad \text{Eq. (4.3)}$$

$$\lambda_2 = 2\beta_1 - \beta_0 \quad \text{Eq. (4.4)}$$

$$\lambda_3 = 6\beta_2 - 6\beta_1 + \beta_0 \quad \text{Eq. (4.5)}$$

$$\lambda_4 = 20\beta_3 - 30\beta_2 + 12\beta_1 - \beta_0 \quad \text{Eq. (4.6)}$$

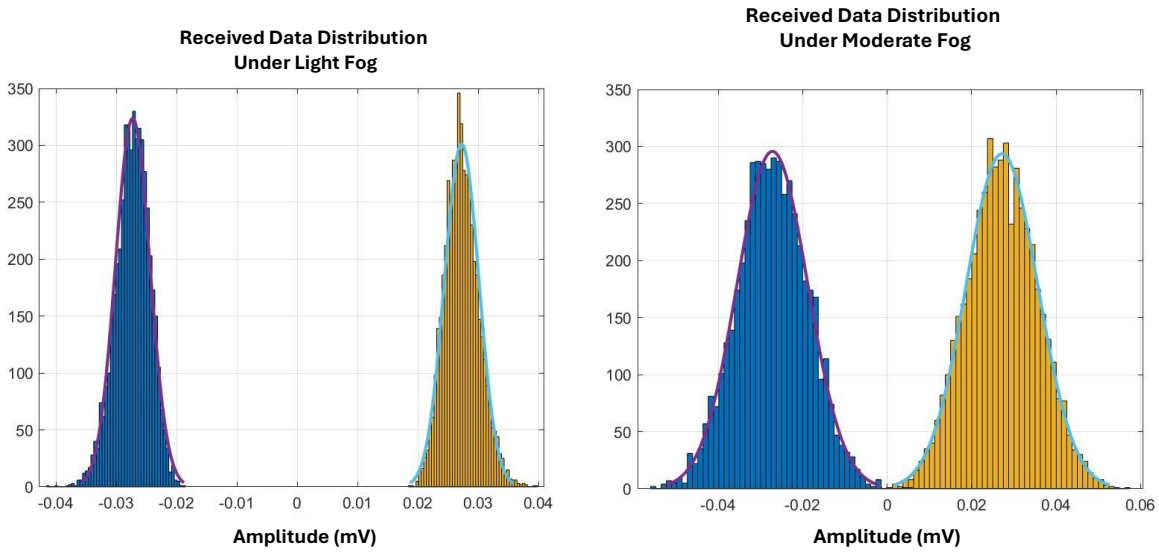
where λ_1 is the L-mean, λ_2 is the L-scale, λ_3 is the L-skewness, and λ_4 is the L-kurtosis.

Numerical analysis confirmed a lognormal distribution of the received data, as given in Eq. (4.7), indicating the presence of multiplicative noise effects commonly observed in turbulent optical channels.

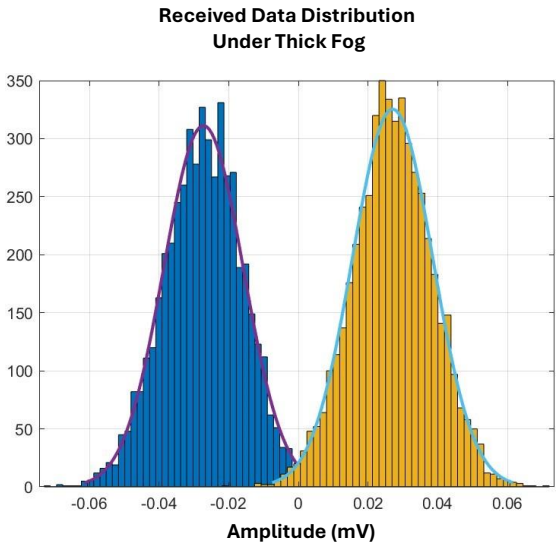
$$PL(d) = PL(d_0) + 20\log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \quad \text{Eq. (4.7)}$$

where $PL(d)$ is the path loss at distance d (dB), $PL(d_0)$ is the path loss at reference distance d_0 (dB), and $X_\sigma$denotes zero-mean Gaussian random variable with standard deviation σ (dB).

Figure 4-7 illustrates the received data distributions along their respective best-fit lognormal curves for light, moderate, and thick fog conditions.



(a) (b)



(c)

Figure 4-7 Received data histogram with best-fit lognormal distribution (a) under light fog, (b) under moderate fog, (c) under thick fog

4.4 Methodology

A deep learning-based (DL-based) decoder was designed to reconstruct the original data stream from a turbulence-affected received signal. The decoder leverages a fully connected feed-forward neural network (NN) structure with layers of linear and ReLU activation functions (see Figure 4-8). Model is optimized using the Adam optimizer, which adapts the learning rate based on first- and second-order moment estimates of the gradients. The training objective is to minimize to minimize the cross-entropy loss function, which quantifies the difference between the predicted and actual output distributions.

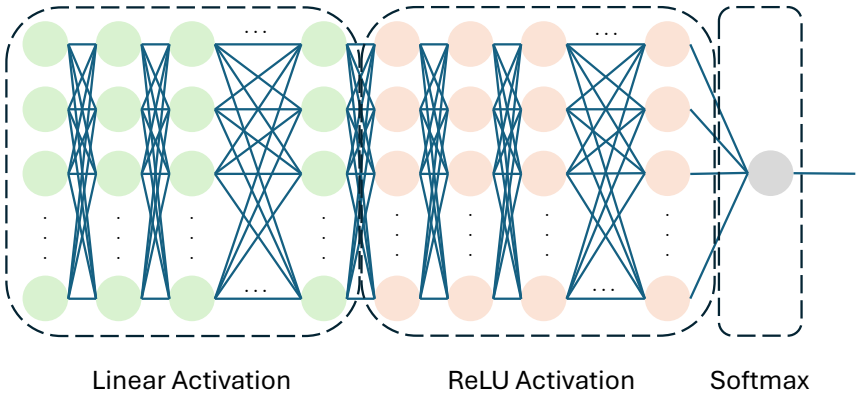


Figure 4-8 Fully connected feed-forward neural network (NN) structure

Decoder architecture comprises of the following components:

1) The input layer gets the received signal, $y \in R^n$, where n is the number of samples in the received signal subjected to atmospheric turbulence.
2) The hidden layers consist of L fully connected layers, each with H_l neurons. The activation functions used in the hidden layers are as follows:
    a. Linear activation for the first few layers to learn simple linear relations.
    b. Rectified linear unit (ReLU) activation for later layers introduces nonlinearity.

Each hidden layer output is given by Eq. (4.8).

$$\boldsymbol{h}_l = \sigma(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l) \quad \text{Eq. (4.8)}$$

where σ is the activation function (linear or ReLU), $\boldsymbol{W}_l \in \mathbb{R}^{H_{l-1} \times H_l}$ represents weight matrix for the l^{th} layer, $\boldsymbol{h}_{l-1}$is the Output from previous layer, and $\boldsymbol{b}_l \in \mathbb{R}^{H_l}$is the bias vector of the l^{th} layer.

3) Output Layer uses softmax activation to map the decoder's output to the probability distribution of the possible classes, producing the final prediction $\hat{y}$ given by Eq. (4.9).

$$\hat{y} = Softmax(\boldsymbol{W}_L \boldsymbol{h}_{L-1} + \boldsymbol{b}_L) \quad \text{Eq. (4.9)}$$

where $\boldsymbol{W}_L \in R^{H_{L-1} \times K}$ is the weight matrix and K is the number of possible classes, and $\boldsymbol{b}_L$is the bias matrix of the output layer.

Key structural hyperparameters include the number of hidden layers L and the number of neurons per layer H_l are adjusted to achieve the required BLER performance.

Training process aims to minimize the cross-entropy loss, a standard metric for classification tasks, which corresponds to maximizing the likelihood of the correct class being predicted18. The loss function is defined by Eq. (4.10).

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}[\, y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \quad \text{Eq. (4.10)}$$

where N in the number of samples and y_i is the true label for the i^{th} sample.

The Adam optimizer used to minimize the loss function by updating model weights. Adam is an adaptive optimizer that calculates individual learning rates based on the estimates of the moments of the gradients [50]. It combines the advantages of both momentum (by tracking past gradients) and RMSProp (by scaling the learning rates based on recent gradients), making it

effective for training deep learning models [50]. The first moment (mean) and second moment (variance) of the gradients are computed using Eq. (4.11) and Eq. (4.12), respectively.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta \mathcal{L}(\theta) \quad \text{Eq. (4.11)}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\left(\nabla_\theta \mathcal{L}(\theta)\right)^2 \quad \text{Eq. (1.12)}$$

where ∇_θ denotes the gradient for the parameters θ (weights and biases), $\beta_1$is the first momentum decay rate, $\beta_2$is the second momentum decay rate. Corrections can be obtained using Eq. (4.13) and Eq. (4.14).

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad \text{Eq. (4.13)}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_t^2} \quad \text{Eq. (4.14)}$$

Finally, parameters are updated using Eq. 15.

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \quad \text{Eq. (4.15)}$$

where ϵ is a small constant added to prevent division by zero.

Hyperparameters for Adam Optimizer have been set to ($\eta = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, batch\ size = 1000,\ epochs = 100$) [51]. Figure 4-9 shows the training losses for different learning rates.

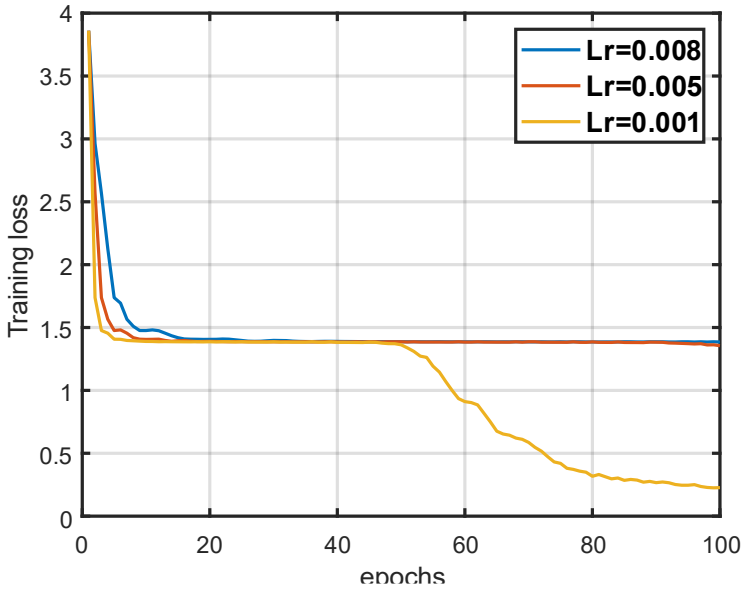


Figure 4-9 Training loss for different learning rates

To prevent overfitting, the training process was monitored using validation loss and accuracy [52]. Training stops when the desired accuracy threshold (90%) is met. Algorithm 1 explains the stopping criteria.

Algorithm 1 Training Stopping Criteria

Input: Training data:D_{train},

Validation data: D_{val},

Desired validation accuracy $\alpha = 90\%$

Output: Trained DL-based decoder

Initialize:

Set initial validation accuracy $val_{acc} = 0$

Set initial validation loss $val_{loss} = \infty$

Train the Model:

While $val_{acc} < \alpha$

Train DL-NN on D_{train}

Compute val_{acc} and new_val_{loss} on D_{val}

If $new\_val_{loss} \geq \alpha$:

Update $val_{loss} = new\_val_{loss}$

Repeat:

Train DL-NN on D_{train}

Compute new_val_{loss} on D_{val}

If $new_{val_{loss}} < val_{loss}$

Update $val_{loss} = new\_val_{loss}$

Else: Stop training

4.5 RESULTS AND DISCUSSION

Figure 4-10 shows the structural hyperparameters of the DL decoder used under both light and moderate fog conditions.

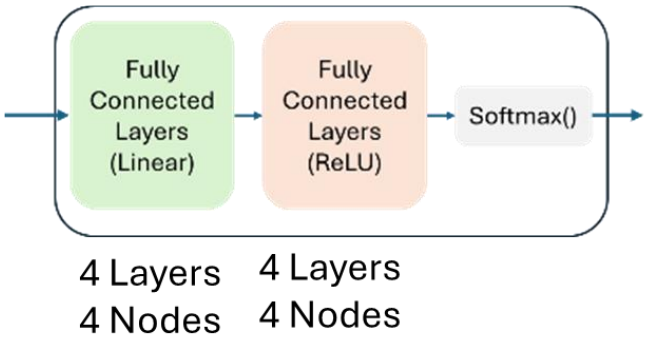


Figure 4-10 DL-Decoder for light and moderate fog

We evaluated the performance of the proposed DL-based decoder against a conventional decoder using BLER as the primary metric. A block was formed using eight consecutive bits. Under light fog conditions (LWC < 0.05), the proposed DL-based decoder achieves a lower BLER than the conventional system for for $Eb\ /N_0$ < 5 dB, with an average performance gain of approximately 0.25 dB. (see Figure Figure 4-11). The improvement is more pronounced for $Eb\ /N_0$ > 5 dB, where the DL decoder consistently reduces BLER by 0.3 dB compared to the conventional system. Similarly, proposed decoder outperformed that of the conventional system under moderate fog conditions achieving average gain of 0.38 dB.

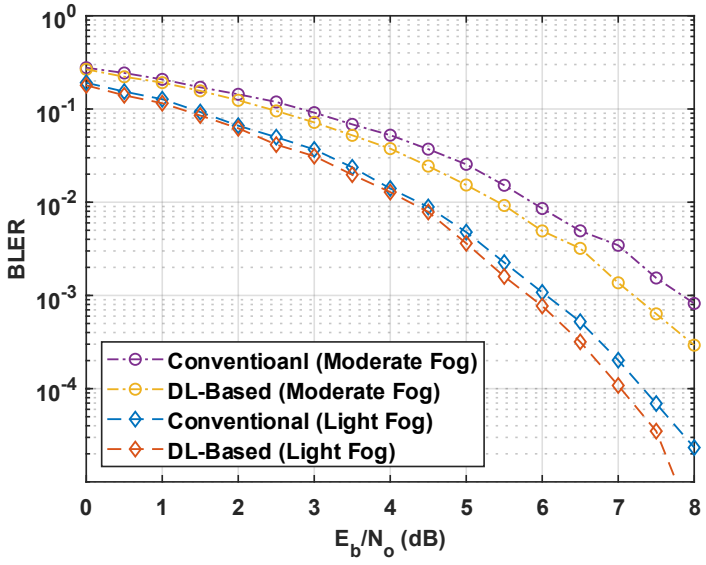


Figure 4-11 BLER vs $E_b\ /N_0$ for light and moderate fog

To prevent overfitting, we monitored validation loss and accuracy during training. Under light fog scenario, validation loss initially decreased but began to rise again after approximately 200 iterations (see Figure 4-12), indicating overfitting. This suggests that the elbow point is a critical

stopping criterion to avoid degradation in model generalization. Under moderate fog conditions, the model required more training iterations to reach the desired performance level before overfitting occurred.

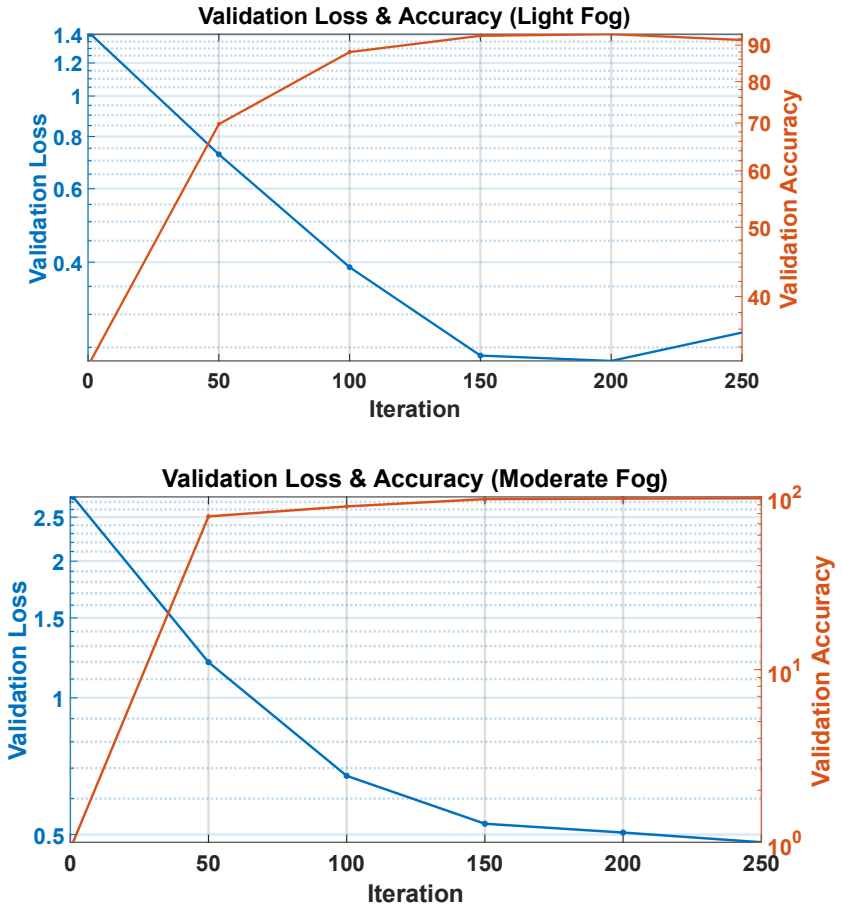


Figure 4-12 Validation loss and accuracy for light and moderate fog

Under thick fog conditions, the proposed structure struggled to achieve the same performance as the conventional system, even with double training iterations (see Figure 4-13).

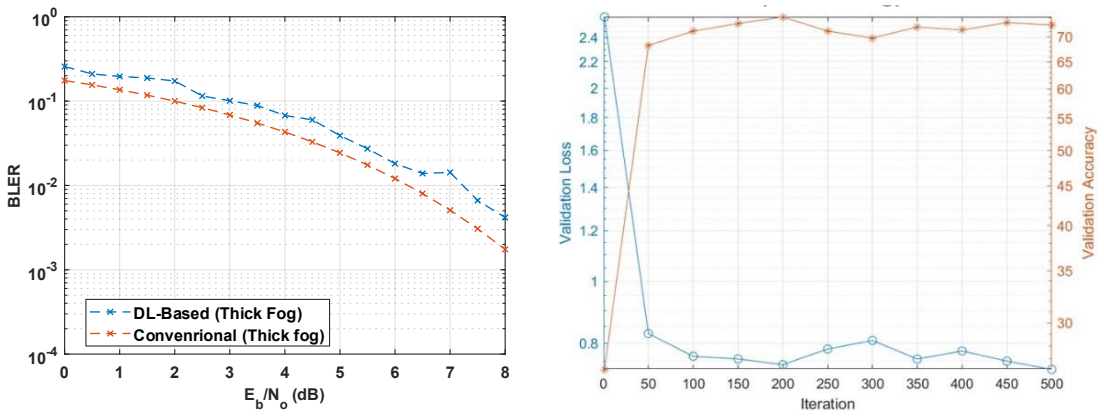


Figure 4-13 BLER vs E_b/N_0 and training performance for thick fog (shallow structure)

To compensate for the increased turbulence effects, we enahanced the network depth by increasing in the number of hidden layers from four to six in each set. Figure 4-14 depicts the modified structure of the decoder used for the thick fog.

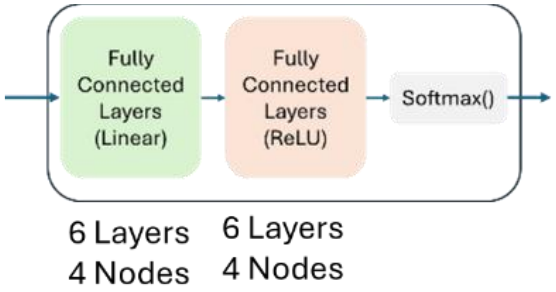


Figure 4-14 DL-Decoder for thick fog

This deeper architecture significantly improved BLER performance, achieving around 1.6 dB gain on average compared to convention system.

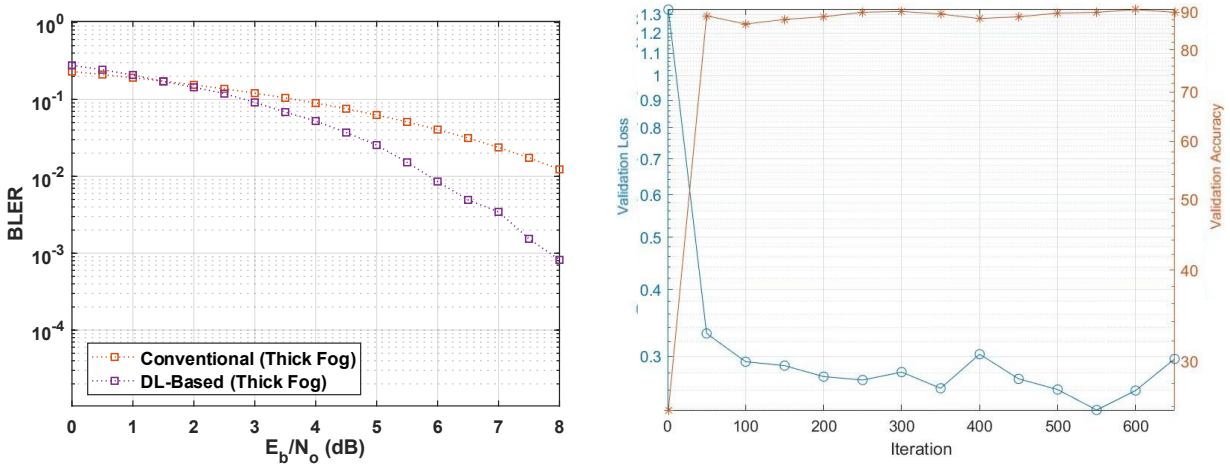


Figure 4-15 BLER vs E_b/N_0 and training performance for thick fog (modified structure)

As poor channel conditions require deeper NN structures, it is critical to quantify the trade - off between performance gain and computational cost. Therefore, we implemented multiple DL-based decoder structures by varying the number of linear and ReLU layers symmetrically. To account for the added computational cost, we calculated the number of floating-point operations (FLOPs) [53] for each structure. Figure 4-16 illustrates the BLER results for each structure under moderate fog conditions and the corresponding FLOPs.

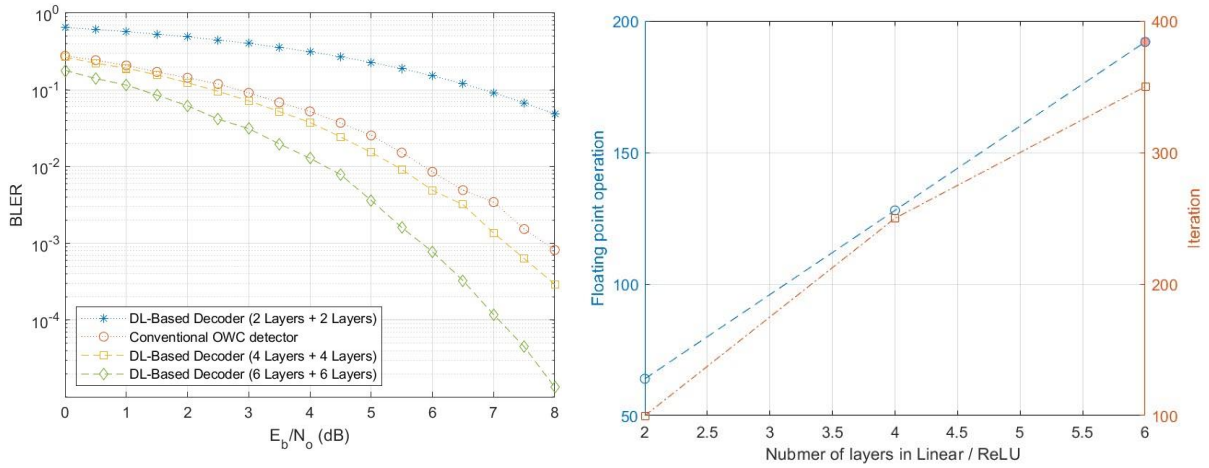


Figure 4-16 BLER and FLOPs for different numbers of deep layers

Evidently, deeper structures add more gain in terms of BLER. For instance, a 6-layer decoder adds 2 dB average gain in comparison to the 4-layer decoder. However, the computational cost grows linearly with the number of layers.

4.6 Conclusion

This study explored enhancing the OWC decoder performance by training a DL-based decoder to reconstruct symbols from a turbulent channel. Unlike most related works, which heavily depend on simulation, empirical data over 1310 nm were collected in a controlled laboratory environment to train the DL-NN.

Using a controlled weather chamber, we collected empirical data at 1310 nm under varying fog conditions in a turbulent environment. Our results confirmed that sufficiently trained feedforward fully connected DL layers consistently score lower BLER than conventional optical detector. However, poor channel conditions require deeper decoders to attain the targeted performance. While increasing the number of hidden layers improves BLER, it also significantly

raises the associated computational complexity and cost, which must be balanced for practical implementation.

The training performance was monitored precisely to avoid overfitting the training dataset. A BLER comparison of different structures confirms that deeper NNs can add gain to BLER on the cost of complexity.

Work is in progress to extend this approach to handle multiuser scenarios, where interference hurdles the signal reconstruction. Additionally, we are investigating adaptive decoders by using dynamic neural networks to allow the model to adjust its structure and parameters to adaptively with varying channel impairments.

5 Simulation, Implementation, and Evaluation of a Deep Learning-Based Wireless Decoder Using Real-World Over-the-Air Data

In Chapter 2, we examined how deep learning techniques have been employed to optimize wireless communication systems, providing a comprehensive literature review on their potential and limitations. Later, Chapter 3 introduced the foundational theory of autoencoders, discussed their architectural variations, and explored recent developments in wireless autoencoder design. These chapters laid the groundwork for implementing and evaluating a practical deep learning-based communication system.

This chapter presents the core experimental work conducted to demonstrate the feasibility and performance of a wireless autoencoder, benchmarking it against conventional OFDM-based systems. The study comprises three major stages. First, we simulated a complete wireless channel autoencoder to evaluate its ability to replicate the functionality of a traditional baseband transceiver. Using fully connected feedforward neural networks, we designed the encoder (transmitter) and decoder (receiver) of a baseband OFDM transceiver tailored for IEEE 802.11n Wi-Fi (see Figure 5-1). We validated the performance by comparing simulated results with theoretical benchmarks using error-based metrics, focusing particularly on the block error rate (BLER).

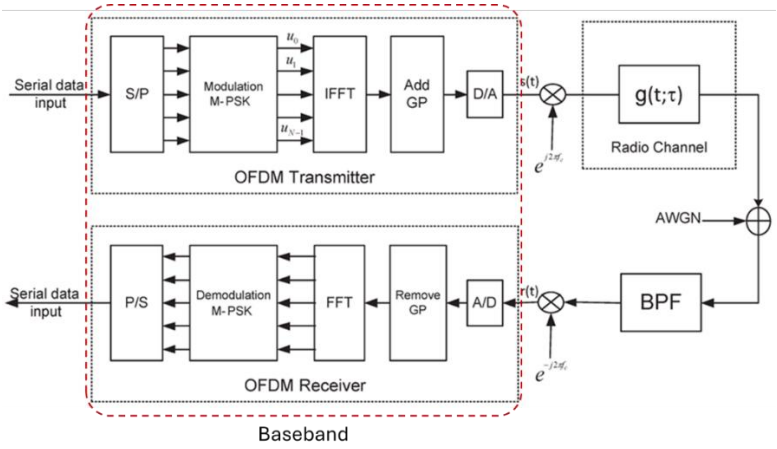


Figure 5-1 Conventional OFDM Transceiver

Secondly, to move beyond idealized simulations, we developed a lab-based experimental platform using software-defined radio (SDR) equipment to collect real-time baseband data. This data was used to train the decoder portion of the autoencoder, enabling it to reconstruct transmitted data samples under realistic channel conditions.

Lastly, we examine the complexity-performance trade-offs associated with deeper neural network structures, which are critical for real-time deployment in resource-constrained devices. Our findings serve as a foundation for future work that aims to extend deep learning-based wireless receivers to support higher-order modulation schemes (e.g., 64-QAM) and integrate with emerging wireless standards.

5.1 Background

Wireless communication systems have historically achieved significant advances in spectral efficiency, reliability, and throughput through mathematically optimized designs. These approaches rely on accurate channel modeling and block-wise optimization, but often fall short when dealing with highly dynamic or complex environments, where model mismatch and computational burden become significant. In contrast, deep learning offers a powerful alternative by learning directly from empirical data without requiring explicit models. Neural networks are capable of capturing complex, nonlinear channel characteristics and generalizing to new conditions, making them ideal for real-time adaptive communication systems.

While most prior research on deep learning-based wireless receivers has been confined to simulations, this chapter contributes to the field by leveraging real-world data captured over-the-air. The SDR-based IEEE 802.11n OFDM transceiver introduces realistic impairments such as RF front-end nonlinearities, thermal noise, carrier frequency offsets, and other hardware-induced distortions that are often abstracted away in simulations. By training the deep learning decoder on this data, we evaluate its ability to operate under practical constraints.

The implemented decoder uses a fully connected feedforward neural network, and we analyze its performance by varying architectural parameters such as the number of hidden layers. We also highlight key observations during training, including occasional loss spikes and accuracy drops,

and discuss potential causes such as overfitting, learning rate instability, and sensitivity to hyperparameter tuning.

5.2 Simulation of OFDM-based wireless autoencoder

This section presents the simulation framework and results of an end-to-end wireless autoencoder designed to perform the functionality of a traditional orthogonal frequency division multiplexing (OFDM) system. The goal is to investigate whether a deep neural network (DNN) can effectively learn the behavior of an OFDM transceiver and potentially achieve improved performance under realistic channel impairments.

5.2.1 OFDM System Setup

To establish a benchmark for comparison, we first implemented a conventional baseband OFDM transceiver in MATLAB. The system supports QPSK / 16-QAM modulations and operates over a baseband additive white Gaussian noise (AWGN) channel. Key parameters used in the simulation are listed in Table 5-1.

Table 5-1 OFDM System Parameters

Number of subcarriers (IFFT Size)	64
MIMO configuration	2x2
Cyclic prefix size	4 samples
Number of bits per symbol	2 / 4
Modulation type	QPSK / 16-QAM
Channel Model	AWGN
E_b/N_0 range	0 - 8

The AWGN channel adds Gaussian-distributed noise to the transmitted signal with zero mean and variance determined by the desired SNR value. For each SNR step, 10,000 OFDM symbols were transmitted and received, ensuring statistically significant performance results. These transmitted-received symbol pairs form the labeled dataset used for training the autoencoder.

5.2.2 Autoencoder Architecture and Training

The autoencoder was designed using fully connected (dense) feedforward neural networks for both the encoder (transmitter) and decoder (receiver) components. The objective is to train the network to reconstruct the original input QPSK symbols from their noisy, channel-corrupted

representations. The high-level structure of the autoencoder is shown in Figure 5-2 and includes the following layers:

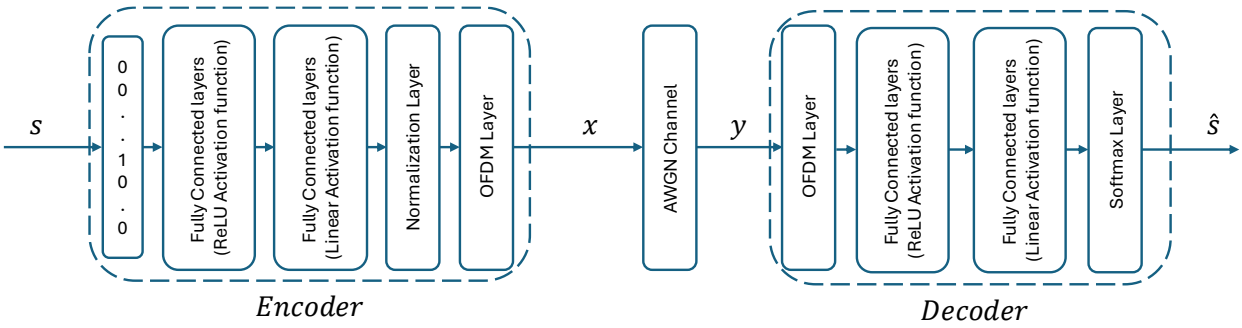


Figure 5-2 Autoencoder structure for OFDM simulation

- Input Layer: One-hot encoded vector representing the QPSK symbol (length = 4)
- Encoder:
    - Dense layers with ReLU activation
    - Dense layers with linear activation
- Normalization Layer: forces average power constraint to $E[|x_i|^2] = 1, \forall\, i$.
- AWGN Channel Layer: Simulated using a custom layer that adds noise according to SNR
- Decoder:
    - Dense layers with ReLU activation
    - Dense layers with linear activation
- Output layer: with softmax activation (to classify back into one of the QPSK / 16-QAM symbols)

The dataset of 10,000 samples was divided using the holdout validation method into 90% training and 10% validation sets. The cross-entropy loss function was used to measure the difference between the original one-hot input vector and the softmax output of the decoder. The model was trained using the Adam optimizer with a learning rate of 0.001, batch size of 128, and trained for 100 epochs.

5.2.3 Simulation Results

Figure 5-3 shows the block error rate (BLER) performance of the trained autoencoder compared to the conventional OFDM system over varying SNR values. Remarkably, the autoencoder achieved lower BLER across the entire tested SNR range, especially at low SNRs (e.g., 0–8 dB), where conventional systems tend to degrade rapidly. This demonstrates the potential of neural networks to learn more robust representations under noisy conditions.

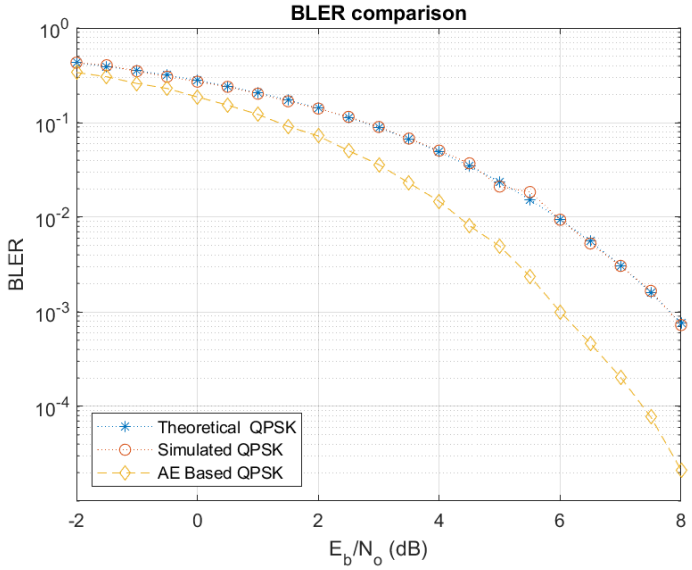


Figure 5-3 BLER versus Eb/N0 for QPSK-OFDM based Autoencoder.

Figure 5-4 displays the validation loss and accuracy curves over training epochs. The network converges steadily, with minimal overfitting observed, indicating appropriate model complexity and effective regularization.

These outcomes align with previous findings in [13], [16], [17], and [18], which support the idea that deep learning-based communication systems can learn compact representations that are inherently more resilient to noise and impairments.

However, when extending the autoencoder framework to support higher-order modulation, specifically 16-QAM, several differences in performance and training behavior were observed. Due to the increased constellation complexity and denser symbol mapping, the autoencoder required significantly more training iterations and careful hyperparameter tuning to achieve performance gains over the conventional OFDM system.

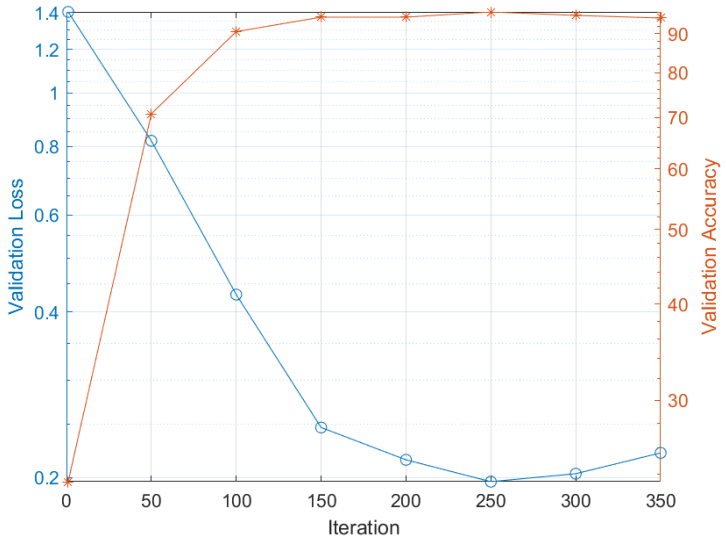


Figure 5-4 Validation Loss and Accuracy versus Training iterations (QPSK-AE)

In early epochs, the model struggled to distinguish between closely spaced constellation points, resulting in higher classification error. Nonetheless, with sufficient training and an appropriate learning rate schedule, the network eventually outperformed the traditional 16-QAM OFDM transceiver, particularly at moderate to high SNRs. This improvement is illustrated in Figure 5-5 and Figure 5-6, which compare the BLER and validation accuracy trends across both systems. These findings emphasize the need for deeper or more expressive neural network architectures and extended training regimes when targeting higher-order modulation formats.

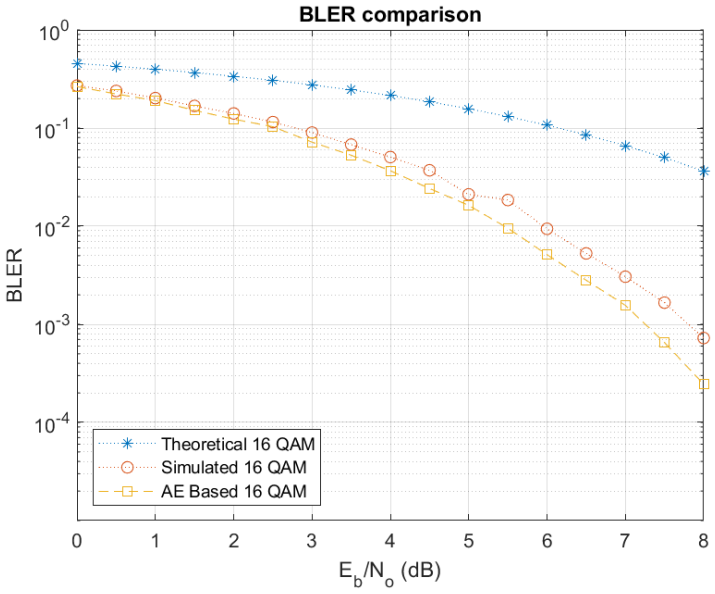


Figure 5-5 BLER versus Eb/N0 for 16-QAM-OFDM based Autoencoder.

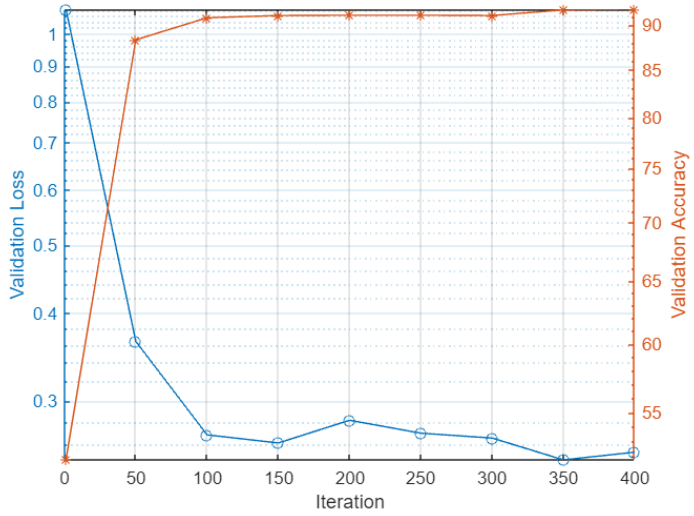
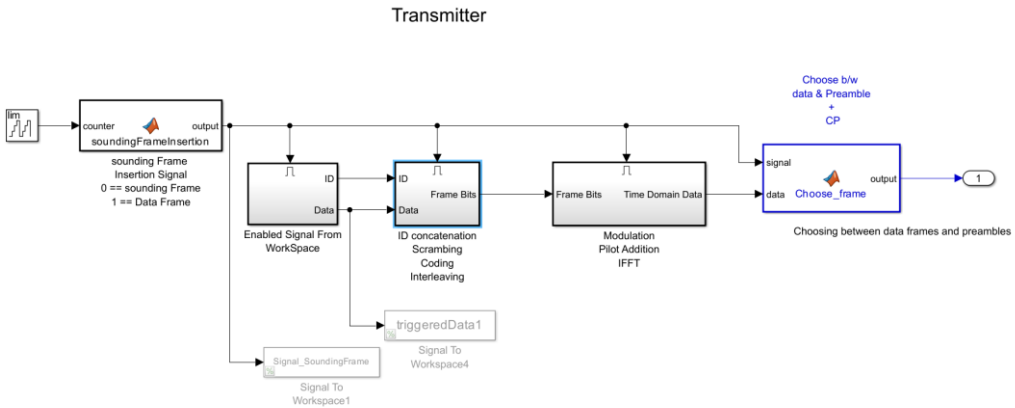


Figure 5-6 Validation Loss and Accuracy versus Training iterations (16-QAM-AE)

5.3 Implementation and Evaluation of a Deep Learning-Based Wireless Decoder Using Real-World Over-the-Air Data

As mentioned, most previous works used simulation only to evaluate the autoencoder in wireless communication. Hence, our work aims to establish a practical framework to collect real-time data to be utilized in wireless autoencoder development.

Software Defined Radio (SDR) is a scalable platform that enables researchers to design and deploy customizable wireless systems. We used Universal Software Radio Peripheral (USRP) X310 to implement an OFDM transceiver, which we simulated earlier. Later, we used this design to collect real-time OFDM data for training purposes. Figure 5-7 displays the SDR transmitter and receiver implemented using MATLAB.



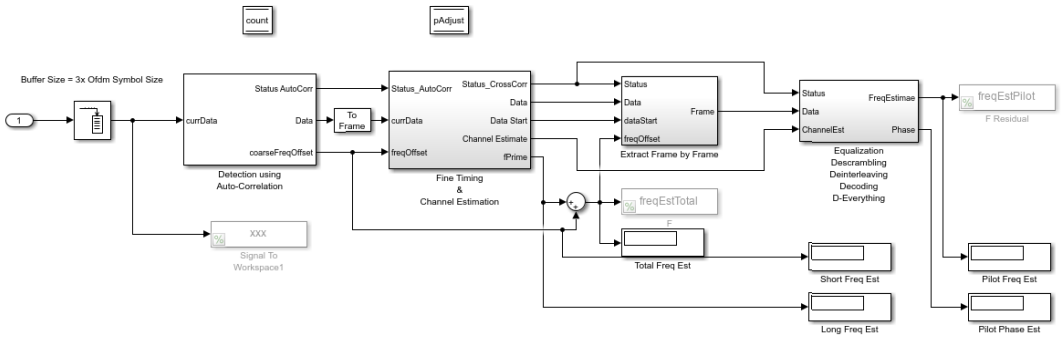


Figure 5-7 SDR Transmitter and Receiver

In fact, at this stage, our goal is to train only the decoder (receiver) portion of the wireless autoencoder. Figure 5-8 shows the block diagram of the OFDM system with an autoencoder receiver. Tests were performed at the Wireless and Electromagnetic Compatibility and Design (WECAD) Center in OU-Tulsa.

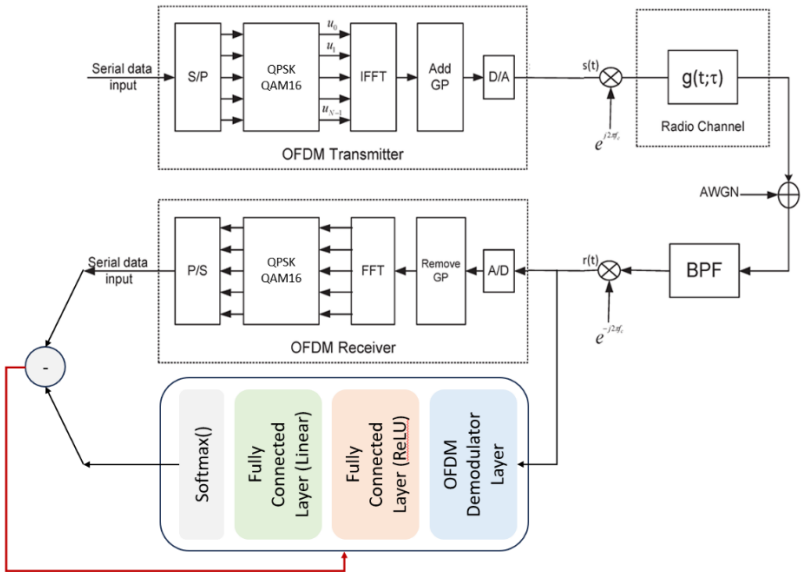


Figure 5-8 OFDM Block Diagram with Autoencoder Receiver

5.3.1 Experimental Setup

This study establishes a practical experimental framework for collecting empirical over-the-air (OTA) data to train and evaluate DL-based wireless models.

The experimental testbed, illustrated in Figure 5-9, comprises two National Instruments (NI) Universal Software Radio Peripheral (USRP) X310 units. One USRP is configured as an OFDM

transmitter, another functions as the receiver. Each unit is equipped with a 6 dBi passive omnidirectional antenna to facilitate wireless transmission and reception in a laboratory setting.

To ensure accurate signal characterization and data integrity, two NI PXIe-1082 vector signal analyzers are deployed one at the transmitter end and one at the receiver. These analyzers capture the RF signal waveforms and monitor key signal metrics such as error vector magnitude (EVM), channel impairments, and noise floor levels throughout the experiment.

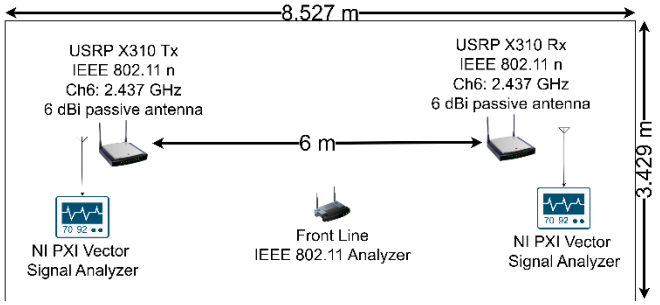


Figure 5-9 Experimental Setup

The OFDM waveform consists of 64 subcarriers with a cyclic prefix length of 16 samples, operating at a 10 Msps sampling rate. Frame synchronization is achieved using a correlation-based method applied to a known preamble, and channel estimation is optionally performed using pilot subcarriers.

A pseudo-random bit sequence (PRBS) of 120 Mbps is generated at the transmitter and modulated using OFDM. The signal is transmitted over Wi-Fi channel 6 (center frequency: 2.437 GHz) with a 20 MHz bandwidth. The received baseband waveform is demodulated and stored for training and evaluating the decoder neural network.

Prior to data collection, an extensive 900-second RF scan is conducted across the 2.4 GHz ISM band to baseline the environmental noise and confirm the absence of external source co-channel or adjacent-channel interference. Figure 5-10 presents the RF power histogram, confirming a stable and controlled wireless environment.

Environmental variability such as multipath reflections and occasional human movement was intentionally left uncontrolled to mimic realistic indoor channel conditions. The experimental

environment includes metallic lab benches, equipment racks, and glass surfaces, contributing to rich multipath propagation.

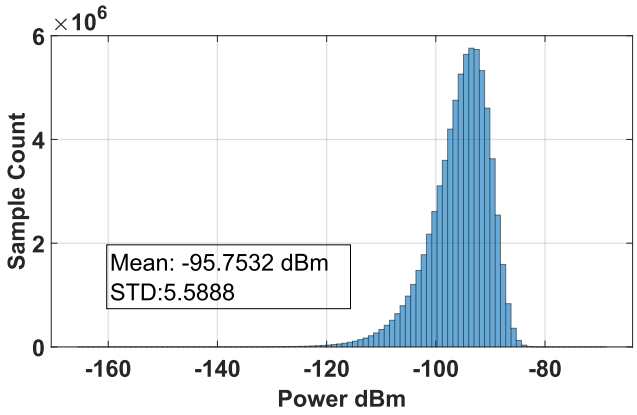


Figure 5-10 Power histogram of lab environment survey

The transmit power was adjusted such that the received power at the receiver input was approximately –50 dBm, as shown in Figure 5-11.

This level was selected to simulate a typical indoor communication scenario with moderate signal strength and to ensure non-saturation of the receiver chain.

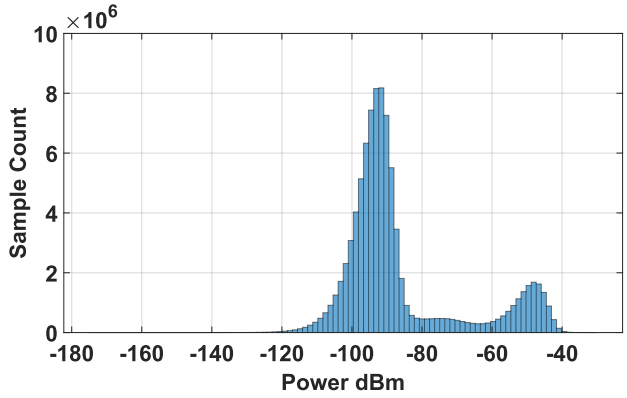


Figure 5-11 Power histogram in presence of Tx signal

The captured OTA signals at the receiver were post-processed and fed into a deep neural network (DNN)-based decoder for evaluation and training. The collected dataset serves as a foundation for developing and validating learning-based wireless receivers under realistic channel conditions that are not easily captured through simulation alone.

5.4 Methodology

A deep learning (DL)-based decoder was developed to reconstruct the original data stream from a received signal affected by channel impairments. The decoder employs a fully connected feedforward neural network (NN) architecture comprising a sequence of linear and non-linear transformations, as illustrated in Figure 5-12.

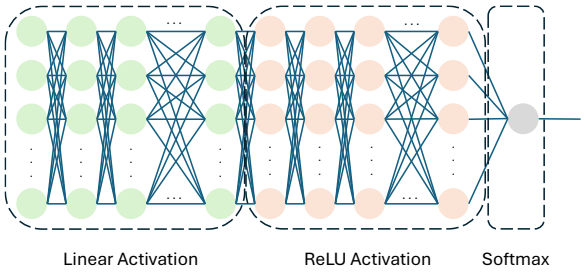


Figure 5-12 Fully connected feed-forward neural network (NN) structure

The decoder is composed of the following key components:

1) Input layer: receives the distorted signal vector $y \in R^n$, where n represents the number of time or frequency domain samples collected at the receiver after experiencing channel impairments.

2) L fully connected hidden layers: each containing H_l neurons. These layers are designed to extract hierarchical features using a combination of:

    a. Linear activation functions in the initial layers to capture basic relationships.

    b. ReLU (Rectified Linear Unit) activation in deeper layers to introduce non-linearity and enable the learning of complex patterns.

Hidden layer output is given by Eq. (5.1).

$$a^{(l)} = f^{(l)}\left(W^{(l)}a^{(l-1)} + b^{(l)}\right) \qquad \text{Eq. (5.1)}$$

where $\mathrm{f}^{(\mathrm{l})}$ is layer activation function, $\mathrm{W}^{(\mathrm{l})} \in \mathbb{R}^{H_{l-1} \times H_l}$ represents weight matrix for the l^{th} layer, $\mathrm{a}^{(\mathrm{l}-1)}$ indicates the activations from the previous layer, and $\mathrm{b}^{(\mathrm{l})} \in \mathbb{R}^{H_l}$ is the bias vector of the l^{th} layer.

3) Output Layer: uses softmax activation to map the decoder's output to the probability distribution of the possible classes, producing the final prediction $\hat{y}$ given by Eq. (5.2).

$$\hat{y} = Softmax\left(W^{(L)}a^{(L-1)} + b^{(L)}\right) \quad \text{Eq. (5.2)}$$

where $W^{(L)} \in R^{H_{L-1} \times K}$ is the weight matrix of output layer and K is the number of possible classes, and $\mathrm{b}^{(\mathrm{L})}$is the bias vector of the output layer.

The decoder is trained using cross-entropy loss, a standard loss function for classification tasks:

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}[\, y_i \, log(\hat{y}_i) + (1 - y_i) \, log(1 - \hat{y}_i) \quad \text{Eq. (5.3)}$$

where N in the number of samples and y_i is the true label for the i^{th} sample.

Model parameters are optimized using the Adam optimizer, which adaptively adjusts learning rates for each parameter using estimates of the first (mean) and second (uncentered variance) moments of the gradients given by Eq. (5.4) and Eq. (5.5), respectively.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta \mathcal{L}(\theta) \quad \text{Eq. (5.4)}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\left(\nabla_\theta \mathcal{L}(\theta)\right)^2 \quad \text{Eq. (5.5)}$$

where ∇_θ denotes the gradient for the parameters θ (weights and biases), $\beta_1$is the first momentum decay rate, $\beta_2$is the second momentum decay rate. The update rules for Adam are given by Eq. (5.6).

$$\theta_t = \theta_{t-1} - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad \text{Eq. (5.6)}$$

where η_t is the learning rate and ϵ is a small constant prevents division by zero. To ensure stable training and convergence, a learning rate decay schedule is applied. Specifically, the learning rate η_t is reduced at regular intervals according to Eq. (5.7).

$$\eta_t = \eta_0 . \gamma^{\left(\frac{1}{T}\right)} \quad \text{Eq. (5.7)}$$

where η_0 is the initial learning rate, γ denotes the decay rate, and $T$is the number of epochs per decay. Hyperparameters for Adam Optimizer have been set to the default values ($\eta_0 = 10^{-3}$,$\beta_1 = 0.9$,$\beta_2 = 0.999$, $\epsilon = 10^{-8}$, batch size=64, epochs=100, $\gamma = 0.5$, and $T = 10$). Figure 5-13 shows the training losses for different learning rates.

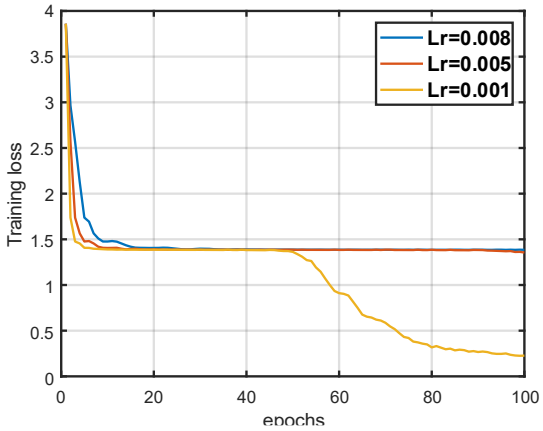


Figure 5-13 Training loss for different learning rates.

To avoid overfitting and ensure acceptable model generalization, early stopping is incorporated into the training process. The model's performance on a validation set is monitored after achieving the minimum threshold of validation accuracy of 90%. If the validation loss does not improve after a certain number of epochs (patience), training is halted early to prevent unnecessary computations and overfitting. The early stopping criterion can be described in algorithm (1).

ALGORITHM 1 TRAINING STOPPING CRITERIA

Input: Training data:D_{train},
Validation data: D_{val},
Desired validation accuracy $\alpha = 90\%$
Output: Trained DL-based decoder
Initialize:
Set initial validation accuracy $val_{acc} = 0$
Set initial validation loss $val_{loss} = \infty$
Train the Model:
While $val_{acc} < \alpha$
Train DL-NN on D_{train}
Compute val_{acc} and new_val_{loss} on D_{val}
If $new\_val_{loss} \geq \alpha$:
Update $val_{loss} = new\_val_{loss}$
Repeat:
Train DL-NN on D_{train}
Compute new_val_{loss} on D_{val}
If $new_{val_{loss}} < val_{loss}$
Update $val_{loss} = new\_val_{loss}$
Else: Stop training

5.4.1 RESULTS AND DISCUSSION

Figure 5-14 shows the structural hyperparameters of the DL decoder.

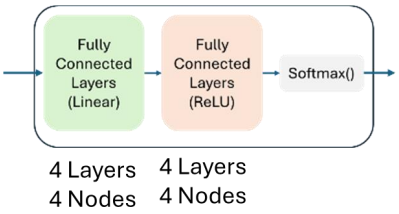


Figure 5-14 DL-Decoder Structure

The performance of the proposed DL-based decoder was evaluated by comparing its BLER with that of a conventional system. A block was formed using eight consecutive bits.

Figure 5-15 demonstrates that the proposed DL-based decoder outperforms the conventional receiver for QPSK modulation. The performance improvement is evident, with the BLER gradually decreasing as the E_b/N_0 increases, showing highest improvements at higher E_b/N_0. This result highlights the efficacy of the DL-decoder in mitigating noise and interference, particularly at higher signal-to-noise ratios (SNR). Our proposed design achieved 3 dB additional gain at $E_b/N_0 = 8\ dB$.

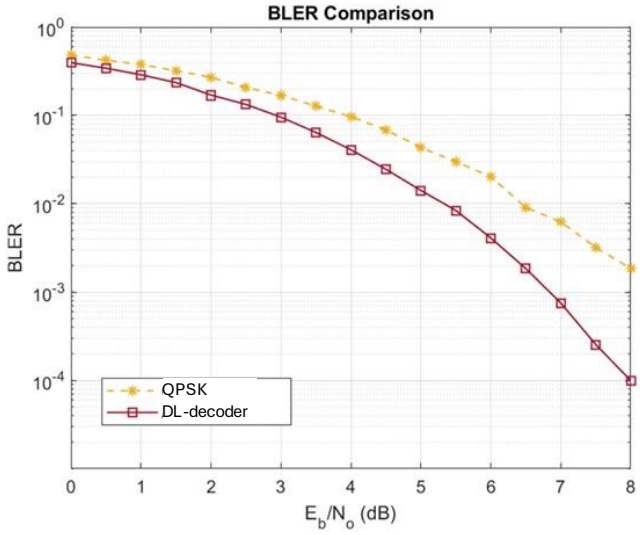


Figure 5-15 BLER performance of DL-Receiver for QPSK

It is crucial to assess the training performance through validation loss and accuracy to estimate the cost and effectiveness of implementing a DL-based decoder. The validation loss serves as an indicator of overfitting, while the validation accuracy provides insight into how well the model generalizes to unseen data. Figure 5-16 shows the validation loss and accuracy for the QPSK DL-based decoder, confirming that the model converges efficiently to a solution.

It is observed that validation loss initially decreased but start to increase after approximately 200 iterations (see Figure 5-16), suggesting overfitting. This advocates that the elbow point is a critical stopping criterion to avoid generalization degradation.

For 16-QAM, the trained DL-decoder initially slightly matched the performance of the conventional system in the E_b/N_0 range of 0 dB to 3 dB.

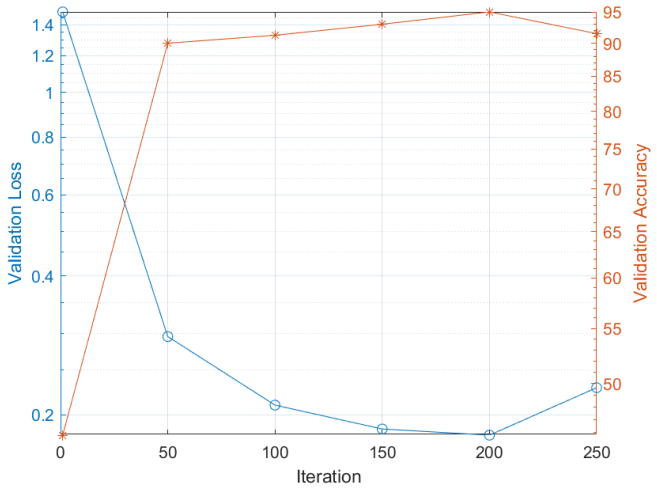


Figure 5-16 Validation loss and accuracy for QPSK DL-based decoder

However, beyond this range, the DL-decoder outperformed the conventional receiver, achieving a lower BLER (see Figure 5-17). This behavior is expected due to the increased probability of error in 16-QAM, where constellation points are closer together, making the system more susceptible to noise and interference [10].

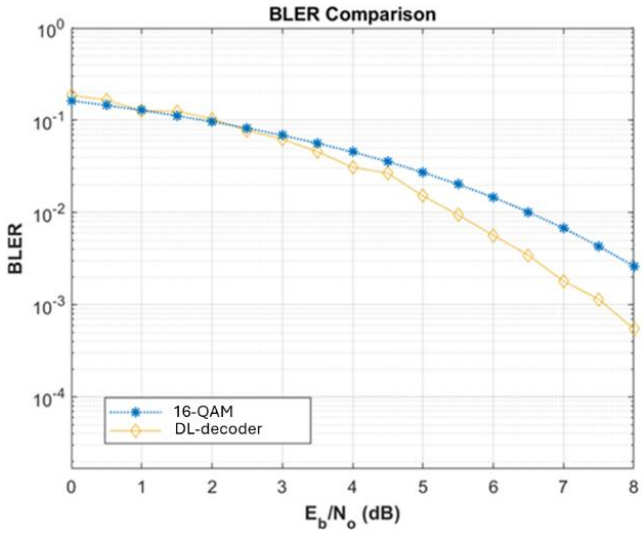


Figure 5-17 BLER performance of DL-Receiver for 16-QAM

While the DL-decoder for 16-QAM achieved superior performance, it required more training iterations than the QPSK decoder, which is due to the more complex decision boundaries in the higher-order modulation scheme. Figure 5-18 depicts the validation loss and accuracy for 16-QAM.

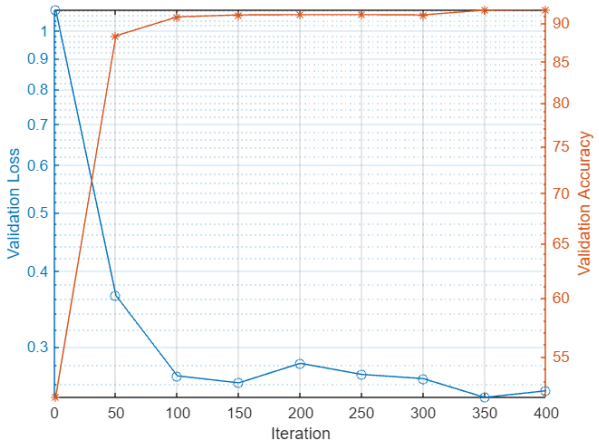


Figure 5-18 Validation loss and accuracy for 16-QAM DL-based decoder

The structure of the neural network (NN) plays a pivotal role in its performance. The number of hidden layers is a critical hyperparameter affecting the decoder's ability to generalize and the overall system performance. We investigated the impact of the depth of the fully connected layers on the QPSK DL-decoder's performance.

Figure 5-19 clearly shows that the deeper the network, the greater the improvement in BLER for QPSK. This improvement demonstrates the network's ability to capture more complex patterns and relationships between the input signal and the target output. However, this performance gain comes at the cost of increased computational complexity and energy consumption, which must be considered for practical implementations.

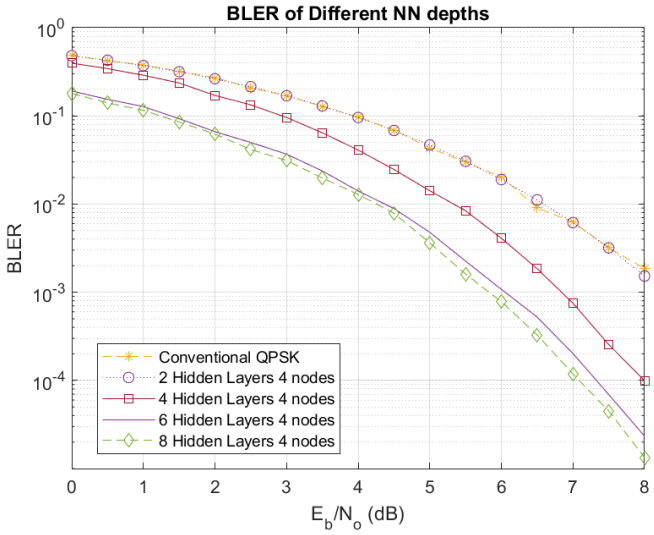


Figure 5-19 BLER for different number of hidden layers in QPSK DL-decoder

To evaluate the additional computational cost, we determined the number of floating-point operations (FLOPs) [17]. Figure 5-20 shows that the computational overhead is increasing linearly as the number of hidden layer increase.

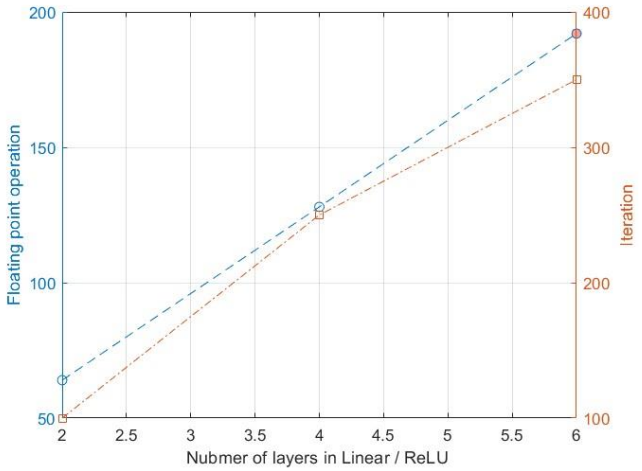


Figure 5-20 FLOPs for different numbers of hidden layers

5.5 Conclusion

This study demonstrated the effectiveness of a deep learning-based decoder for enhancing the performance of wireless communication systems.

The proposed DL-decoder outperforms conventional receivers in terms of BLER across different modulation schemes, including QPSK and 16-QAM. The results show that the decoder requires more training iterations for complex constellations. Additionally, we explored the impact

of network depth on performance, highlighting a trade-off between improved accuracy and increased computational complexity.

Our approach also integrates practical data collection using SDR, ensuring that the trained model can be applied to real-world scenarios with varying channel conditions. The use of a well-structured experimental setup, combined with early stopping and other optimization techniques, further strengthens the reliability and efficiency of the model. Overall, this work paves the way for deploying deep learning-based decoders in dynamic and interference-prone communication environments, offering a promising alternative to traditional systems for improving communication reliability and robustness. Future work will focus on refining the decoder architecture and extending it to handle multi-user interference scenarios for more comprehensive system performance evaluation.

6 Toward AI Native Wireless Communication Physical Layer Using Dynamic Deep Neural Networks

6.1 Introduction

Adaptive modulation schemes have become a cornerstone of modern wireless communication systems, enabling transceivers to dynamically select modulation orders based on prevailing channel conditions. This adaptability enhances spectral efficiency, throughput, and robustness by striking a balance between data rate and error resilience. In standards such as IEEE 802.11n, modulation schemes like QPSK, 16-QAM, and 64-QAM are employed adaptively to optimize performance in response to time-varying wireless environments.

Deep learning (DL) has recently emerged as a powerful tool for physical-layer signal processing, offering data-driven approaches to receiver design that can outperform traditional model-based techniques in complex or non-ideal settings. Prior work including our earlier study on deep learning-based decoders—demonstrated that neural networks (NNs) can effectively learn to decode orthogonal frequency-division multiplexing (OFDM) symbols from over-the-air baseband samples, capturing hardware impairments and channel distortions that are difficult to model analytically.

However, conventional deep neural network (CDNN) decoders are typically static in nature: once trained, their architecture and parameters remain fixed during inference [54]. This rigidity limits their efficiency and adaptability when deployed in environments with varying computational constraints and modulation schemes. For instance, decoding high-order modulations like 64-QAM requires deeper networks with higher computational complexity, while lower-order schemes such as QPSK can be accurately decoded with shallower or simplified models.

To address this gap, we propose a dynamic neural decoder architecture tailored for adaptive modulation in IEEE 802.11n systems. Dynamic neural networks adapt their structures and/or parameters to enhance efficiency, computational cost, and generality to various applications [55], [56]. Classical approach suggests constructing model ensemble using cascaded or parallel

substructures that can be activated conditionally depending on the input [57], [58]. However, dynamic networks have the flexibility to allocate different computational resources (layers, nodes, sun-networks, or parameters) at the inference time to improve system performance [59], [60].

Our approach enables the decoder to dynamically adjust its computational graph or parameters based on the detected modulation order. We investigate three dynamic strategies:

1. Modulation-aware network switching: Separate neural networks are trained for QPSK, 16-QAM, and 64-QAM, and the appropriate network is activated based on the incoming constellation.
2. Early-exit architecture: A unified deep network is trained for 64-QAM, with intermediate exit points that allow early inference for simpler modulation orders, thereby reducing computation.
3. Dynamic weight reloading: A single architecture is trained with different sets of parameters tailored to each modulation order. During inference, the model adapts by reloading the appropriate weights.

Each dynamic strategy is evaluated in terms of block error rate (BLER), energy efficiency, and floating-point operation (FLOP) count, highlighting the trade-offs between accuracy and complexity. Our results show that:

- The modulation-aware switching network achieves near-optimal decoding performance with minimal complexity per constellation.
- The early-exit model significantly reduces FLOPs for lower-order modulations without sacrificing decoding accuracy.
- Dynamic weight reloading provides a hardware-friendly compromise between memory overhead and computational savings.

These findings demonstrate the feasibility and advantages of deploying dynamic neural network decoders for practical adaptive wireless systems, especially in edge devices with strict latency or power constraints.

6.2 Dynamic Neural Networks

To enable input-dependent processing in wireless communication tasks, we designed dynamic neural networks with two main objectives. First, to dynamically adjust network architecture

based on the characteristics of each input, allowing the system to allocate just enough computational resources per sample and thereby eliminate redundant processing [61]. Second, to adapt the network's parameters (weights) while keeping the overall architecture fixed, enhancing the model's representational capacity with only a minimal increase in computational overhead [61]. These strategies are particularly valuable for real-time decoding under varying modulation schemes, where balancing performance and efficiency is critical.

6.3 Dynamic DL-Based Decoder Architecture:

Given that different modulation schemes impose varying levels of decoding complexity, we employ dynamic neural network architectures that adapt their computational effort based on the modulation order. Lower-order modulation schemes such as QPSK are more resilient to noise and channel distortions due to their wider symbol spacing, resulting in lower error probability and requiring less representational power during decoding [62]. In contrast, higher-order schemes such as 16-QAM and 64-QAM pack more bits per symbol at the expense of denser constellations, making them more susceptible to noise and inter-symbol interference [62]. As a result, decoding higher-order modulations typically demands deeper networks, more neurons, and greater numerical precision to accurately distinguish between closely spaced symbols [63].

To address this variation in computational demand, dynamic neural networks can be designed to adjust their depth or inference path based on the detected modulation order. For example, a QPSK input might require only a few shallow layers to achieve reliable decoding, whereas a 64-QAM input would activate the full network to capture the finer decision boundaries. This modulation-aware adaptation conserves energy and computation during favorable channel conditions or when low-complexity modulations are used, while still delivering high accuracy when more challenging constellations are present.

Moreover, traditional static architectures process all input, regardless of their modulation complexity, with the same fixed computational graph [64]. This not only wastes resources on "easy" inputs (e.g., QPSK symbols) but also limits the system's ability to scale efficiently across a range of operating conditions. In contrast, dynamic networks provide conditional computation, allocating resources on demand per input sample, which leads to significant improvements in

energy efficiency, latency, and overall system scalability [61]. Such flexibility is especially beneficial for wireless systems where modulation orders vary in response to real-time channel feedback and link adaptation protocols.

6.3.1 Modulation-aware network switching

In wireless communication systems using adaptive modulation, the complexity of signal decoding varies significantly depending on the modulation scheme (e.g., QPSK, 16-QAM, 64-QAM) [62][65]. To address this, we implemented modulation-aware network switching, a dynamic inference strategy where a controller selects the most suitable neural network sub-module conditioned on the detected modulation format. This approach avoids unnecessary computation by activating only the required network branch, allowing for efficient and scalable decoding across modulation schemes [64].

We modeled this process as sample-wise dynamic routing through a multi-branch architecture, where each branch is a dedicated decoder trained for a specific modulation order. Let the set of available decoders be:

$$\mathcal{N} = \{N_1, N_2, N_3\} \qquad \text{Eq. (6.1)}$$

corresponding to QPSK, 16-QAM, and 64-QAM decoders respectively.

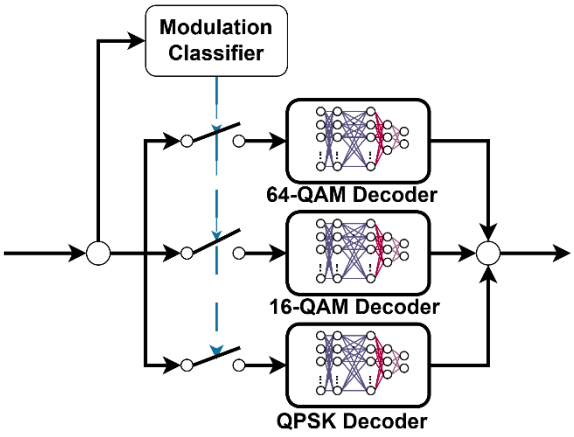


Figure 6-1 Modulation-aware network switching

Let x represent the received signal, and $m \in \{1,2,3\}$ denote the modulation type of the input (inferred via a lightweight modulation classifier). The controller function $f_c(x)$ produces a one-hot routing decision.

$$s = f_c(x) \in \{e_1, e_2, e_3\} \quad \text{Eq. (6.2)}$$

where e_i is a one-hot vector activating the decoder N_i. The final output is given by Eq. (6.3):

$$\hat{y} = \sum_{i=1}^{3} s_i \, . N_i(x) \quad \text{Eq. (6.3)}$$

Since s is a one-hot vector, only one decoder is active per inference, significantly reducing computation compared to monolithic models that process all inputs with a fixed architecture.

This mechanism is analogous to a hard mixture-of-experts (MoE) model [59] without fusion where each decoder (expert) handles a distinct modulation constellation, and no aggregation is performed across branches. This form of modulation-dependent inference offers a tailored trade-off between decoding complexity and representation capacity, conditioned on the error sensitivity and symbol density of each modulation order. For instance:

- **QPSK**, with high symbol spacing and low BER, requires shallow, low-complexity networks.
- **64-QAM**, with dense constellations and higher BER under noise, requires deeper, more expressive architectures.

This modulation-aware switching strategy belongs to a broader family of conditional computation models, where each input is routed through a customized computation path. It ensures low latency and energy efficiency for "easier" modulation types, while maintaining high accuracy for complex signals, outperforming static models that allocate equal computation regardless of modulation difficulty.

6.3.2 Early Exit Architecture

While modulation-aware network switching achieves adaptability by selecting a dedicated sub-network for each modulation scheme, this approach requires maintaining and managing multiple distinct models, which may increase memory overhead and deployment complexity especially in resource-constrained wireless systems. As a complementary strategy, we explored dynamic depth architecture early-exit architectures that leverage a single unified network shared across all modulation orders, with the ability to dynamically adjust its inference depth based on input complexity.

This mechanism is known as dynamic depth control, and it operates by strategically skipping the execution of deeper layers when early layers produce confident outputs [66]. The core idea is to equip the neural decoder with intermediate exit points positioned at specific layers in the network. These exits consist of lightweight modulation classifier that detect the modulation scheme of the incoming constellation points and activating the layer skipping gates accordingly. In implementation, early exits are supported using gating functions [67].

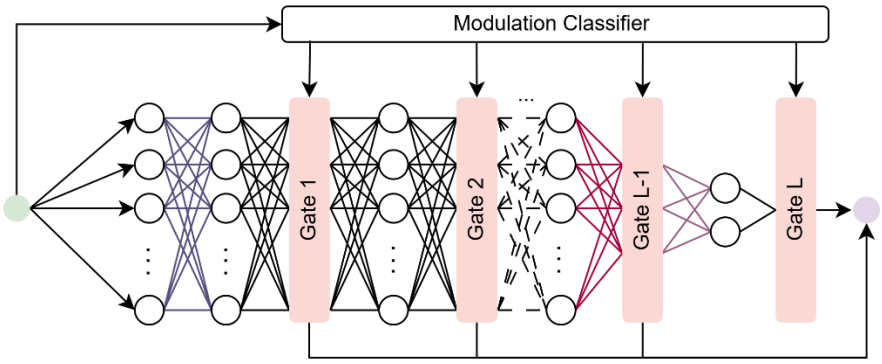


Figure 6-2 Early Exit Architecture

Mathematically, let the input sample be denoted by x_i , and let the neural network consist of L layers $\{F_1, F_2, \dots, F_L\}$. For each input, the network computes intermediate features as follows:

$$h_1 = F_1(x_i), h_2 = F_2(h_1), \dots, h_L = F_L(h_{L-1}) \qquad \text{Eq. (6.4)}$$

Gating function $G_k(h_k) \in \{0,1\}$ is controlled by fully connected modulation classifier to activate or skip deeper layers:

$$x_{k+1} = G_k(h_k).F_k(h_k) + \big(1 - G_k(h_k)\big).h_k \qquad \text{Eq. (6.5)}$$

This structure can be viewed as a conditional computation graph, where each input dynamically selects its own inference depth [68]. It is particularly advantageous for decoding schemes with varying complexity:

- **QPSK samples** often exit at shallow layers due to high symbol separability.
- **64-QAM samples**, being more ambiguous, typically require full-depth inference to maintain accuracy.

Overall, early-exit architectures offer a principled way to match the network's inference complexity to the decoding task's inherent difficulty, achieving significant savings in latency, energy, and floating-point operations without compromising performance on higher-order modulations.

6.3.3 Dynamic Parameters

While dynamic architectures (e.g., early exits or network switching) enable inference path adaptation per input, they often require specially designed structures and tailored training regimes. An alternative modular approach is to maintain a fixed network architecture and dynamically reload parameters based on the incoming modulation type. This technique, referred to as dynamic weight reloading, preserves architectural simplicity while enhancing the decoder's ability to handle a wide range of modulation schemes with minimal computational overhead [69].

In a conventional static decoder, the output for an input x is defined as:

$$y = F(x; \theta) \qquad \text{Eq. (6.6)}$$

where F is the fixed network function and θ denotes the set of static, pre-trained weights.

In contrast, dynamic weight reloading allows the network parameters to vary based on modulation order or channel condition. This can be modeled as:

$$y = F(x; \theta_m) \qquad \text{Eq. (6.7)}$$

where $\theta_m = \mathcal{W}(m)$ are modulation-specific weights that retrieve the appropriate weight configuration for modulation scheme $m \in \{QPSK, 16-QAM, 64-QAM\}$.

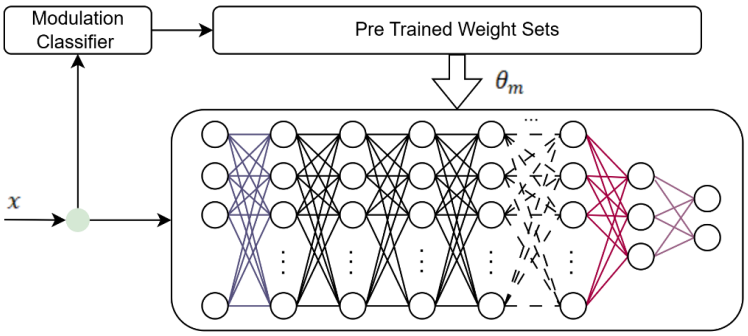


Figure 6-3 Dynamic Parameters Architecture

This formulation ensures that:

- The decoder uses lightweight reconfiguration at runtime (parameter switching), avoiding full retraining or architectural changes.
- Each weight set θ_m is optimized specifically for a modulation scheme, improving representation precision and robustness for that constellation.
- During inference, only the relevant parameters are loaded into the model, enabling fast adaptation to varying modulation conditions in a real-time system.

From a practical perspective, this approach is well-suited for embedded or low-latency environments, as it allows one trained network structure to serve multiple purposes by reloading only small memory blocks [70]. In wireless systems, modulation-aware weight reloading offers a flexible middle ground: more efficient than training multiple full models (as in network switching), and more accurate than training a single generalized decoder.

Additionally, this concept aligns with ideas in meta-learning and conditional computation. In future implementations, the reloading function $\mathcal{W}(m)$ could be extended to depend not only on the modulation type but also on estimated channel state information (CSI) or SNR, enabling finer-grained adaptation to channel impairments.

Overall, dynamic weight reloading provides a modular mechanism to enable adaptive decoding in modulation-varying environments, allowing a shared decoder to generalize across diverse modulation schemes while maintaining high efficiency and low latency.

6.3.4 Experimental Setup

This work builds upon a practical and real-world experimental framework used in chapter 5 to collect over-the-air (OTA) data, enabling the design and evaluation of modulation-aware dynamic neural network decoders under realistic wireless channel conditions.

The experimental setup, illustrated in Figure 6-4, uses two National Instruments (NI) USRP X310 software-defined radios. One USRP operates as an IEEE 802.11n-compliant OFDM transmitter, while the other functions as the receiver. Each USRP is equipped with a 6 dBi omnidirectional antenna, enabling wireless transmission and reception within an indoor lab environment.

To monitor signal quality and maintain data fidelity, the system includes two NI PXIe-1082 vector signal analyzers—one at the transmitter and one at the receiver. These devices provide visibility into key RF metrics, including noise floor, and channel impairment statistics throughout data collection.

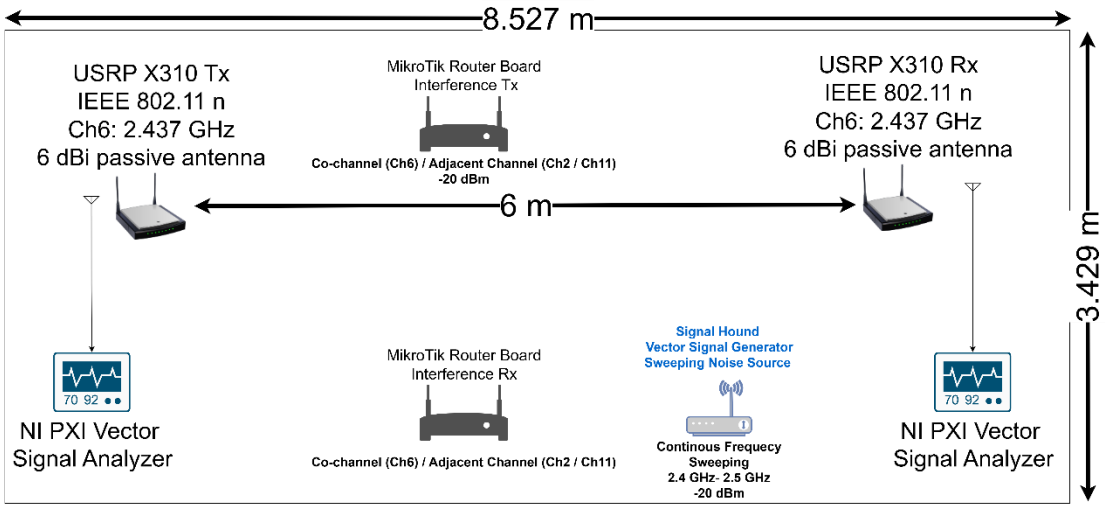


Figure 6-4 Experimental Testbed for Dynamic NN Decoder Evaluation

The transmitted OFDM signal consists of 64 subcarriers, a cyclic prefix of 16 samples, and a sampling rate of 10 Msps. Frame synchronization is performed using a correlation-based method on a known preamble, and pilot subcarriers are used for optional channel estimation. The transmitter generates a pseudo-random bit sequence (PRBS) at 120 Mbps, which is modulated according to one of three modulation schemes—QPSK, 16-QAM, or 64-QAM—and mapped onto OFDM symbols.

The signal is transmitted over Wi-Fi channel 6 (center frequency: 2.437 GHz) with a 20 MHz bandwidth. At the receiver, the demodulated baseband I/Q waveform is captured, labeled according to its modulation type, and stored for decoder training and evaluation.

The resulting dataset includes OTA signals encoded with multiple modulation schemes under real indoor conditions. These signals are used to train and evaluate the proposed modulation-aware dynamic neural network decoder, allowing the study to explore adaptive inference strategies—such as network switching, early exits, and dynamic weight reloading—across varying modulation complexities.

6.3.5 Methodology

For each dynamic structure, the A deep learning (DL)-based decoders were developed and trained to reconstruct the original data stream from a received signal affected by channel impairments. In network switching approach we constructed a separate decoder per modulation type. On the other hand, for the early exit and dynamic parameter reloading we utilized one DL-NN for all modulation types and dynamicity was implemented using gates or weights loading units controlled by modulation classifier. However, we used fully connected feedforward neural network architecture for decoding. This architecture comprises a sequence of linear and non-linear transformations.

The proposed decoder is designed as a modulation-aware feedforward neural network capable of adaptively decoding IEEE 802.11n OFDM signals. It is important to note that while the overall training strategy and loss function remain consistent. Nevertheless, the structural configuration of the neural network varies across the proposed dynamic approaches—namely, modulation-aware network switching, early-exit architecture, and dynamic weight reloading. Each design is tailored to meet specific performance requirements such as block error rate and computational efficiency under different modulation schemes. Accordingly, parameters such as the number of hidden layers and number of nodes are customized per approach. The detailed architectural choices will be presented in the corresponding sections later.

6.3.6 Architecture Overview

The decoder processes the received, distorted baseband signal vector $y \in \mathbb{R}^n$, where n corresponds to the number of frequency-domain subcarrier samples collected after channel distortion and noise. The neural network is composed of the following core components:

- Input Layer: Accepts the raw I/Q samples corresponding to an OFDM frame.
- Hidden Layers: The core of the network comprises L fully connected (dense) hidden layers. Each layer l contains H_l neurons and produces activations $a^{(l)}$according to:

$$\boldsymbol{a}^{(l)} = f^{(l)}\left(\boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}\right) \quad \text{Eq. (6.8)}$$

where $\boldsymbol{W}^{(l)} \in \mathbb{R}^{H_{l-1} \times H_l}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{H_l}$ denote the weight matix and bias vector for layer l and $f^{(l)}$ is the activation function Linear activations are used in earlier layers to preserve signal structure, while ReLU activations are applied in deeper layers to introduce nonlinearity and improve representational capacity.

- Output Layer: The final layer maps the learned features to a probability distribution over K class labels (e.g., bits, symbols, or constellation points) using the softmax function:

$$\hat{y} = Softmax\left(W^{(L)} a^{(L-1)} + b^{(L)}\right) \quad \text{Eq. (6.9)}$$

where $W^{(L)} \in R^{H_{L-1} \times K}$ is the weight matrix of output layer and K is the number of possible classes, and $\mathrm{b}^{(\mathrm{L})}$is the bias vector of the output layer.

The network is trained as a supervised classifier using the **categorical cross-entropy loss**, defined as:

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}[\, y_i \, log(\hat{y}_i) + (1 - y_i) \, log(1 - \hat{y}_i) \quad \text{Eq. (6.10)}$$

where N is the number of training samples, y is the true label vector, and $\hat{y}$ is the predicted output.

Model parameters are optimized using the **Adam optimizer**, which adaptively updates learning rates for each parameter based on first and second-order moment estimates of the gradient. The update rules are:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta \mathcal{L}(\theta) \quad \text{Eq. (6.11)}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\left(\nabla_\theta \mathcal{L}(\theta)\right)^2 \quad \text{Eq. (6.12)}$$

$$\theta_t = \theta_{t-1} - \eta_t \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad \text{Eq. (6.13)}$$

where θ represents the model parameters (weights and biases), $\widehat{m}_t$ and $\hat{v}_t$are bias-corrected moment estimates, and ϵ is a small constant to prevent division by zero.

To ensure stable convergence, a **learning rate decay schedule** is applied:

$$\eta_t = \eta_0 . \gamma^{\left(\frac{1}{T}\right)} \quad \text{Eq. (6.14)}$$

where η_0 is the initial learning rate, γ is the decay factor, and T is the number of epochs per decay step.

The training is configured with the following hyperparameters:

- Initial learning rate: $\eta_0 = 10^{-3}$
- Momentum decay rates: $\beta_1 = 0.9$, $\beta_2 = 0.999$
- Small constant: $\epsilon = 10^{-8}$
- Batch size: 64
- Number of epochs: 100
- Learning rate decay factor: $\gamma = 0.5$, applied every 10 epochs.

Figure 6-5 presents the training loss curves under various learning rate settings, demonstrating the stability and convergence behavior of the model across training runs.

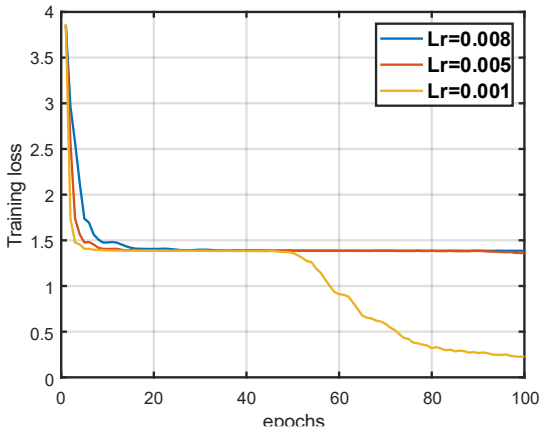


Figure 6-5 Learning rate Vs Epochs

6.3.7 Training Strategies for Each Dynamic Inference Method

6.3.7.1 Modulation-Aware Network Switching

Each sub-network is trained **independently** on data corresponding to a specific modulation scheme (e.g., QPSK, 16-QAM, 64-QAM). The training process for each decoder is isolated, allowing the network to specialize in the characteristics and noise tolerance of its target constellation.

- **Training Schedule:**
  Three networks are trained with identical architectures but independent weights and data subsets.
- **Optimizer Settings:**
  Same hyperparameters are applied across all branches.
- **Loss Monitoring:**
  Training and validation loss are tracked separately for each decoder.
- **Deployment Implication:**
  During inference, a lightweight modulation classifier selects which decoder to activate based on the input constellation.

6.3.7.2 Early-Exit Architecture with Modulation-Guided Gating

In the early-exit strategy, the decoder is structured as a deep feedforward network with multiple internal exit points embedded at predefined layers. However, unlike traditional confidence-based early-exit methods, our architecture integrates a modulation-aware gating mechanism to control the inference depth dynamically. A lightweight modulation classifier predicts the modulation order (QPSK, 16-QAM, or 64-QAM) at the receiver and activates the corresponding exit gate within the network.

Exit Mapping:

- QPSK: Early exit at gate 1
- 16-QAM: Exit at gate 2
- 64-QAM: Full depth inference, no early exit

The gating signal is determined from the output of the modulation classifier M(x)M(x)M(x), and the network output is selected accordingly:

$$\hat{y} = E_{G_m}(x) \qquad \text{Eq. (6.15)}$$

where G_m is the gate index for modulation m.

- **Training Schedule:**
  The network is trained end-to-end with modulation-specific supervision at each exit. Training data is labeled with both the class label (for decoding) and the corresponding

modulation type (for gating). Each exit point is trained on its relevant modulation subset to specialize the intermediate features accordingly.

- **Loss Function:**
  The total training loss is a weighted sum of the cross-entropy losses at each active exit:
  $$\mathcal{L}_{total} = \sum_m \mathcal{L}_{E_{G_m}} \quad \text{Eq. (6.16)}$$

- **Gating Efficiency:**
  Since we are not using confidence based exit method, the exit decision is deterministic, which offers a predictable and efficient complexity based on known modulation characteristics.

- **Implementation Benefit:**
  This approach avoids the computational overhead of dynamic confidence estimation during inference and ensures pre-trained, modulation-optimized exit paths for each modulation type.

This modulation-guided early-exit architecture enables fine-grained control over computational cost while maintaining high decoding accuracy across varying modulation complexities, making it particularly suitable for systems with strict latency and energy constraints.

6.3.7.3 Dynamic Weight Reloading

A single shared network architecture is used, but it is trained with multiple sets of weights, one per modulation scheme. During training, modulation-specific weight sets θ_m are updated only when samples from their corresponding modulation class are presented.

- **Training Schedule:**
  Training alternates between modulation types using a **modulation-aware mini-batch scheduler**, ensuring balanced updates for each weight set.

- **Weight Management:**
  Each θ_m is isolated during updates but shares the same optimizer and architecture.

- **Inference Procedure:**
  Based on detected modulation, the appropriate θ_m is loaded into the network before inference.

6.3.8 Modulation Classifier Design

To enable **modulation-aware dynamic inference**, we incorporate a lightweight **modulation classifier** that predicts the modulation order (e.g., QPSK, 16-QAM, 64-QAM) from received signal

samples prior to decoding. This classifier plays a central role in two of the dynamic strategies: **modulation-aware network switching** and **early-exit gating**, by guiding the selection of the appropriate decoding path.

6.3.8.1 Architecture

The modulation classifier is implemented as a **fully connected feedforward neural network**, composed of:

- An input layer that receives the same preprocessed signal vector $y \in \mathbb{R}^n$ as the decoder.
- L_c hidden layers, each with $H_l^{(c)}$ neurons and ReLU activations.
- An output layer with **softmax activation** over M classes, where $M = 3$ for the three supported modulation schemes: QPSK, 16-QAM, and 64-QAM.

The hidden layer output is defined as:

$$\boldsymbol{a}^{(l)} = \mathrm{ReLU}\left(\boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}\right), 1 \le l \le L_c \qquad \text{Eq. (6.17)}$$

The final output is computed via softmax:

$$\hat{\boldsymbol{p}} = \mathrm{Softmax}(\boldsymbol{W}^{(L_c)}\boldsymbol{a}^{(L_c)} + \boldsymbol{b}^{(L_c)}) \qquad \text{Eq. (6.18)}$$

where $\hat{\boldsymbol{p}} \in \mathbb{R}^{\boldsymbol{3}}$ contains the class probabilities for each modulation scheme.

6.3.8.2 Inference and Integration

The predicted modulation label $\hat{m}$ is selected as:

$$\hat{m} = \arg\max_{j} \hat{p}_j \,, j \in \{1,2,3\} \qquad \text{Eq. (6.19)}$$

This predicted modulation $\hat{m}$ is then used to:

- **Activate the corresponding decoder** in the modulation-aware network switching strategy.
- **Control the gating mechanism** in the early-exit architecture, determining the exit depth.

6.3.8.3 Training

The modulation classifier is trained separately from the decoder using a standard **categorical cross-entropy loss**:

$$\mathcal{L}_{mod} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log\left(\hat{p}_{ij}\right) \quad \text{Eq. (6.20)}$$

where y_{ij} is the one-hot encoded ground truth label for the i_{th} sample and class j, and $\hat{p}_{ij}$ is the predicted probability.

This lightweight classifier converges quickly due to the clear separability of modulation schemes in the signal domain and introduces minimal overhead while enabling **accurate and efficient routing decisions** in dynamic decoder architectures. Figure 6-6 illustrates the confusion matrix of modulation classifier.

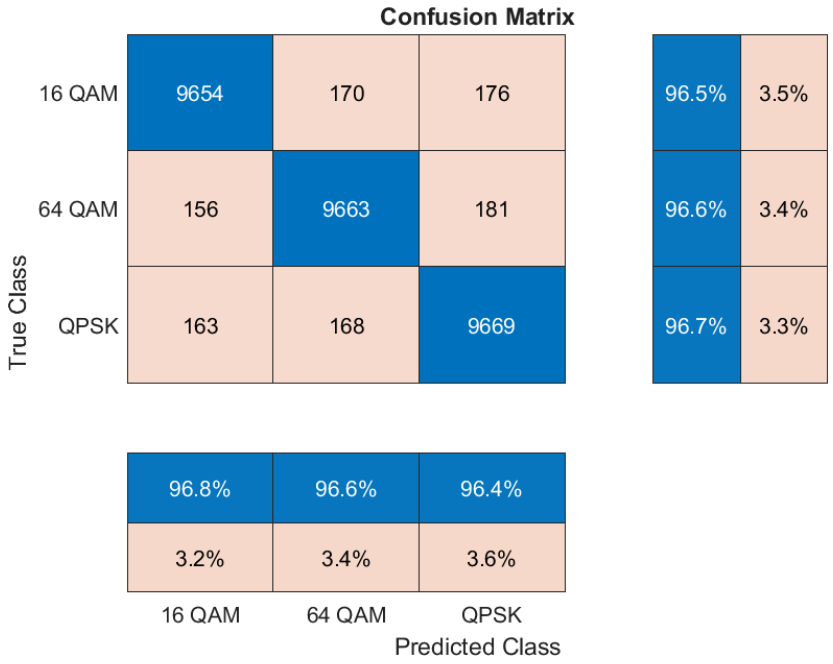


Figure 6-6 Modulation Classifier Confusion Matrix

6.3.9 Results and Discussion:

To evaluate the real-time operation of the proposed dynamic decoding strategies, the system was tested using a practical over-the-air setup. A **USRP X310 software-defined radio (SDR)** was configured as a conventional OFDM transmitter, operating according to the IEEE 802.11n standard. The transmitter was programmed with a **scheduled adaptive modulation scheme**, allowing it to cyclically switch between **QPSK, 16-QAM, and 64-QAM** during the transmission of data frames. This adaptive modulation emulates realistic wireless link behavior under varying channel conditions.

On the receiver side, the corresponding **dynamic neural network decoder** was deployed and connected to a second USRP X310 SDR. The received baseband signals were first processed by a **modulation classifier**, which accurately identified the current modulation scheme from the received waveform. Based on the classifier output, the decoder **dynamically adjusted its structure** at runtime: either by selecting the appropriate sub-network (in the case of modulation-aware switching), activating the correct exit point (in early-exit architectures), or loading the modulation-specific weight configuration (in dynamic weight reloading).

During the operating phase, a **Signal Hound vector signal generator** was used to inject **sweeping noise** across the operational frequency band to range the SNR between 0 and 25 dB. This external noise stimulus forced the transmitter configured with adaptive modulation logic to switch between different modulation and coding schemes (MCS), specifically **MCS 1 (QPSK), MCS 3 (16-QAM), and MCS 5 (64-QAM)**.

This setup validated the feasibility of operating dynamic decoders in a **modulation-varying real-time environment**, demonstrating the system's ability to adapt its inference path or parameters on a per-frame basis with minimal latency and without human intervention. The successful integration of **modulation classification and dynamic inference** confirms the practical viability of the proposed architectures for deployment in future intelligent wireless receivers.

Figure 6-7 shows the structural hyperparameters of the modulation aware network switching architecture.

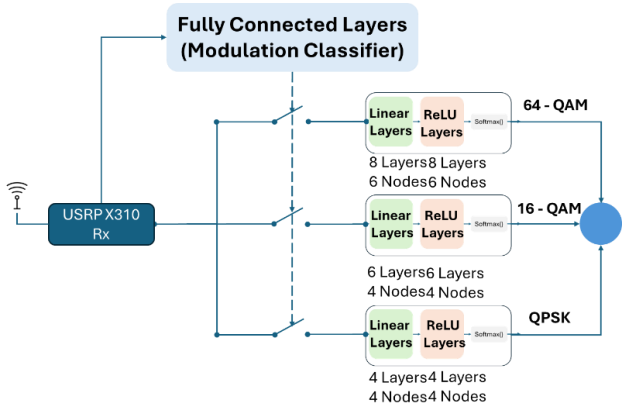


Figure 6-7 Network Switching Decoder Structure

The performance of the proposed DL-based decoder was evaluated by comparing its BLER with that of a conventional system. A block was formed using eight consecutive bits.

Figure 6-8 demonstrates that the proposed DL-based decoders outperform the conventional receivers for QPSK, 16-QAM and 64 QAM modulations.

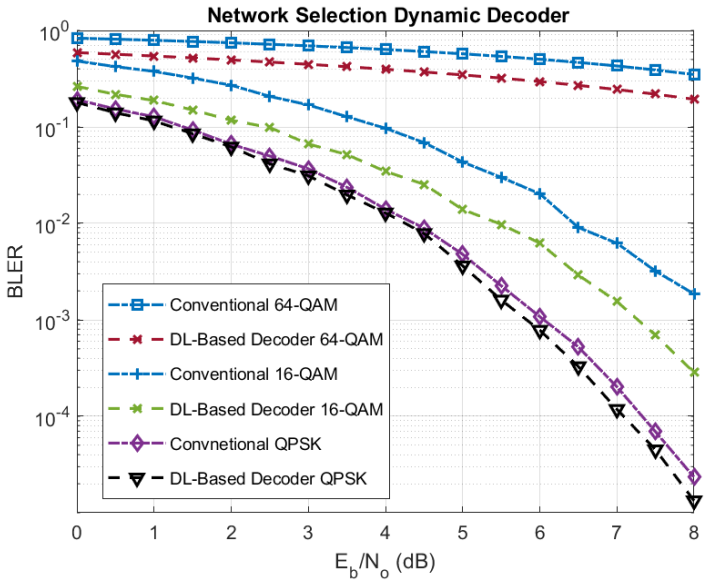


Figure 6-8 BLER Performance of Network Switching Decoder

Figure 6-9 displays the structure of early exit dynamic decoder, where the modulation classifier drives the gates based on the identified modulation scheme.

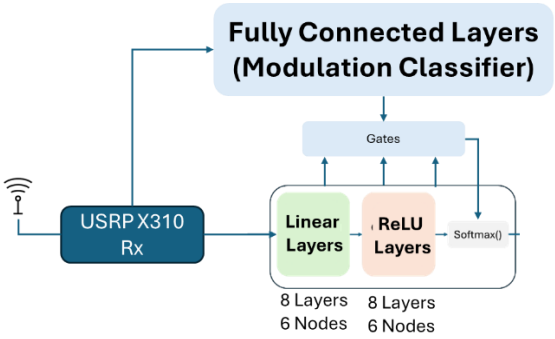


Figure 6-9 Early Exit Decoder Structure

Experimental results revealed that for QPSK, the decoder required processing through all initial linear layers, with an early exit occurring after the second ReLU-activated layer. In the case of 16-QAM, reliable decoding was achieved with an exit at the fourth ReLU layer, while for 64-QAM, the

decoder utilized the entire network depth, completing full forward propagation through all layers. Figure 6-10 depicts the BLER performance for the early exit decoder.

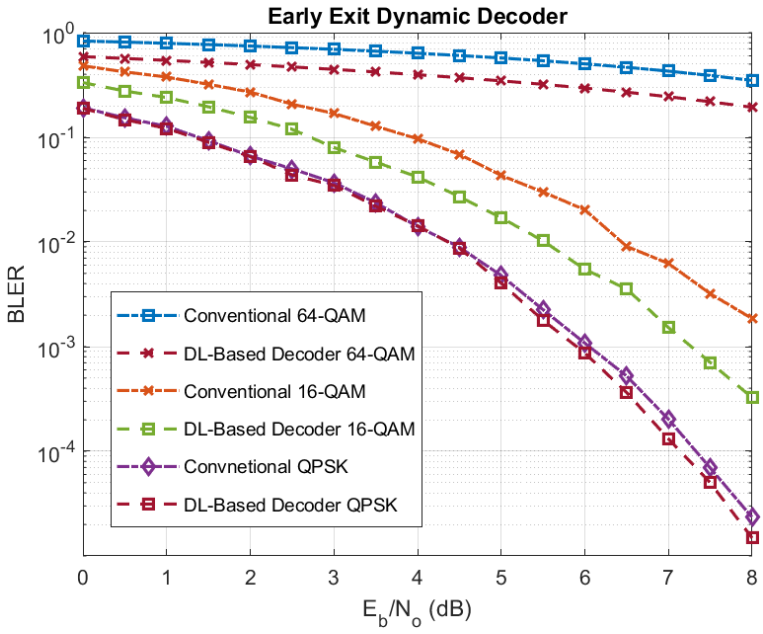


Figure 6-10 BLER Performance of Early Exit Decoder

For dynamic parameters reloading decoder, the same structure is utilized for different modulation schemes, but the weight where loaded based on the identified modulation scheme. (See Figure 6-11).

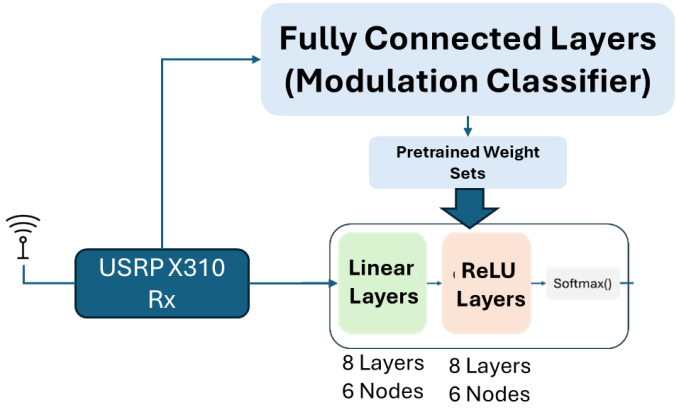


Figure 6-11 Dynamic Parameters Reloading Structure

Similarly, well trained networks were able to outperform the conventional receiver in terms of BLER as shown in Figure 6-12.

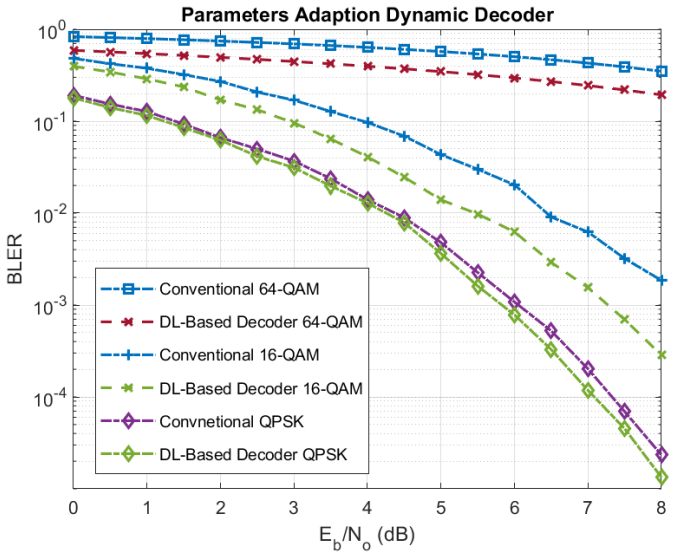


Figure 6-12 BLER Performance of Parameters Reloading Decoder

To quantify the computational efficiency of the proposed dynamic decoding strategies, we evaluated each architecture in terms of **energy consumption** and **floating-point operations (FLOPs)** per inference. These two metrics serve as practical indicators of computational cost—FLOPs provide a hardware-independent measure of the total operations required, while energy consumption reflects the real-world power efficiency of the decoder implementations. For each dynamic strategy (modulation-aware switching, early exit, and dynamic weight reloading), we measured FLOPs based on the number of activated layers and parameters used during inference, conditioned on the modulation order. Correspondingly, energy consumption was estimated using hardware profiling tools and normalized per decoded frame. Energy consumption is primarily associated with:

- Floating Point Operation.
- Activations.
- Data movement: Fetching weights, biases, and activations from memory.
- Special operations: Such as pooling and normalization.

$$Energy\ for\ FLOPs = Total\ FLOPs \times Energy\ per\ FLOP\ (20\ pJ) \quad \text{Eq. (6.21)}$$

$$Energy\ for\ memory = Memory\ accesses \times Energy\ per\ SRAM\ access\ (5\ pJ) \quad \text{Eq. (6.22)}$$

This analysis enables a fair comparison across architectures by revealing how each design trades off complexity for performance, particularly when decoding lower-order modulations that allow early termination or reduced parameter loading.

Table 6-1 Complexity and Energy Consumption of Dynamic NN Structures

Modulation	Strategy	Exit Depth / Structure	FLOPs	Energy (μJ)	Trainable Parameters
QPSK	Early Exit	Exit after 2nd ReLU layer	12,400	0.48	1,180
	Modulation-Aware Switching	4 Linear + 4 ReLU (4 nodes)	14,200	0.52	656
	Dynamic Weight Reloading	Full network (8×6 nodes)	22,600	0.55	3,540 (active: 1,180)
16-QAM	Early Exit	Exit after 4th ReLU layer	19,200	0.81	1,180
	Modulation-Aware Switching	6 Linear + 6 ReLU (4 nodes)	21,500	0.85	696
	Dynamic Weight Reloading	Full network (8×6 nodes)	22,600	0.89	3,540 (active: 1,180)
64-QAM	Early Exit	Full network (8×6 nodes)	29,800	1.32	1,180
	Modulation-Aware Switching	8 Linear + 8 ReLU (6 nodes)	31,000	1.38	1,180
	Dynamic Weight Reloading	Full network (8×6 nodes)	22,600	1.41	3,540 (active: 1,180)

The evaluation shows that early-exit architectures offer the most efficient inference for lower-order modulations like QPSK and 16-QAM, reducing FLOPs and energy consumption by up to 58% compared to full inference. For QPSK, exiting after just two ReLU layers achieved significant computational savings without degrading performance. In contrast, modulation-aware switching, while slightly heavier in FLOPs for low-order modulations, maintains a smaller memory footprint by tailoring a minimal model per modulation.

The dynamic weight reloading strategy, although using the same decoder architecture across all modulations, incurs a larger memory cost due to storing three complete weight sets (3,540 parameters in total). However, its runtime computation remains identical to the early-exit model,

making it a practical option for systems where architecture uniformity is preferred over storage efficiency.

For 64-QAM, all strategies converge in computational demand, as the full network depth is necessary for accurate decoding. This validates the role of dynamic architectures in optimizing resources for easier modulations, while maintaining robustness for higher-order schemes.

Overall, the results highlight the effectiveness of dynamic deep learning strategies in achieving a scalable trade-off between accuracy, complexity, and energy efficiency across diverse modulation schemes.

6.4 Dynamic DL-Based Encoder:

Building on the dynamic decoder architectures presented in the previous section, this section shifts the focus to the transmitter side of the IEEE 802.11n system, where similar principles of dynamic inference can be applied to the deep learning-based encoder. In conventional wireless systems, the transmitter maps incoming data streams to symbols based on a selected modulation scheme, such as QPSK, 16-QAM, or 64-QAM. While adaptive modulation is already employed at the PHY layer to adjust data rates according to channel conditions, traditional encoding architectures remain static, with fixed computational paths regardless of the complexity of the modulation in use.

In contrast, a dynamic neural network-based encoder enables the transmitter to adapt its internal structure depending on the active modulation order, thereby achieving improved efficiency and resource utilization. Inspired by the modular structure of dynamic decoders, this work explores the design and implementation of modulation-aware dynamic deep encoders, which learn to map binary input sequences directly to modulation-specific constellation symbols using separate neural network branches.

Specifically, we construct three independent neural network encoders, each optimized for a distinct modulation scheme: QPSK (MCS 1), 16-QAM (MCS 3), and 64-QAM (MCS 5). Each encoder is designed using a neural architecture search process to tailor its depth and width to the complexity required by its corresponding constellation. During operation, a modulation-aware

switching mechanism activates the appropriate encoder based on the current modulation setting. This approach ensures that lower-order modulations do not incur the computational overhead of higher-order constellations, enabling significant reductions in complexity and energy consumption.

To assess the benefits of this approach, we evaluate the performance of the dynamic encoder architectures using the same SDR-based IEEE 802.11n testbed introduced in earlier experiments. The encoder's effectiveness is quantified based on three metrics: block error rate (BLER), floating-point operations (FLOPs), and estimated energy consumption per transmitted frame. These results are then compared to a static, shared neural encoder to illustrate the trade-offs between performance, complexity, and adaptivity.

By extending dynamic deep learning strategies to the transmitter side, this chapter demonstrates the potential for a fully end-to-end adaptive neural transceiver, where both encoding and decoding paths dynamically reconfigure themselves according to real-time channel conditions and modulation demands. Such systems promise to improve energy efficiency, reduce latency, and maintain robustness in diverse wireless environments, paving the way toward intelligent and flexible communication systems (see Figure 6-13).

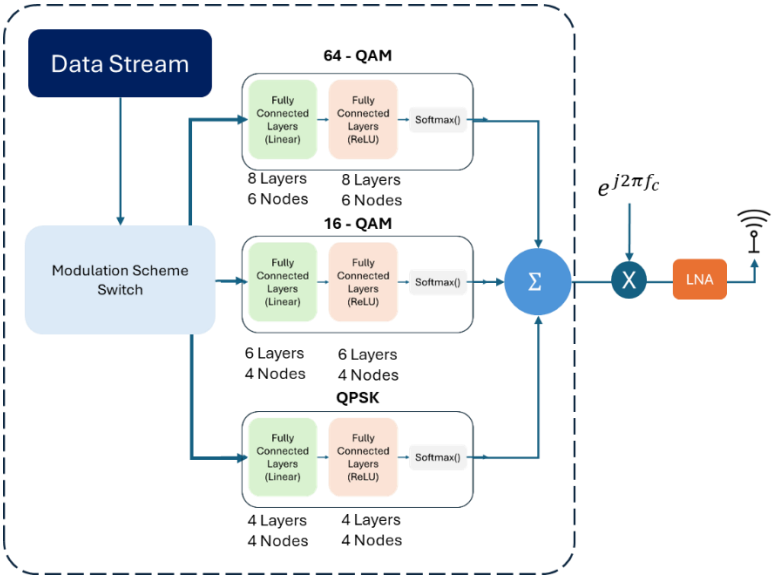


Figure 6-13 Dynamic Encoder using Network Switching

6.4.1 Experimental Setup for Encoder Training

To train and evaluate the proposed dynamic neural encoder architecture, a practical over-the-air experimental setup was developed using commercial RF instrumentation. The objective of the setup is to generate real IEEE 802.11n-compliant constellation points for various modulation schemes, and use them as target references to train a neural network that learns to approximate the modulation process directly from input bitstreams. The setup allows for the precise comparison between standard-generated and DL-generated symbols under ideal conditions, thereby establishing a supervised learning framework for encoder training.

The core component of the testbed is the **Signal Hound VSG60A vector signal generator**, which serves as a high-fidelity, standards-compliant transmitter. The VSG60A is configured to operate using the IEEE 802.11n physical layer stack, and is capable of modulating raw data sequences using one of the supported modulation and coding schemes (MCS): **QPSK (MCS 1), 16-QAM (MCS 3), and 64-QAM (MCS 5)**. For each modulation type, a predefined stream of binary data is encoded and mapped to constellation points according to the IEEE 802.11n specifications.

These **generated constellation points** are captured and stored, along with their corresponding binary input sequences. The captured pairs of {input bits, constellation point} are then used to train a **fully connected feedforward neural network**, where the network learns to map bit sequences directly to I/Q symbol coordinates. Separate models are trained for each modulation scheme, resulting in three distinct encoder networks optimized to reproduce the statistical properties and structure of the QPSK, 16-QAM, and 64-QAM constellations, respectively.

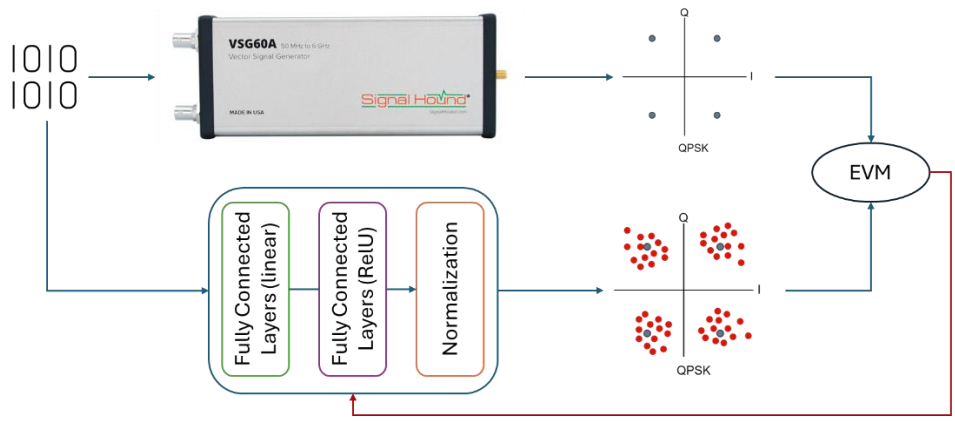


Figure 6-14 Diagram of Constellation Generation and Encoder Training

To quantitatively evaluate the performance of each trained encoder, we use the **Error Vector Magnitude (EVM)** metric. EVM measures the deviation between the ideal (VSG-generated) constellation point and the output of the neural encoder. It is defined as the root-mean-square (RMS) error between the reference and generated constellation symbols, normalized to the average symbol magnitude.

$$EVM_{RMS}[\%] = \sqrt{\frac{\sum_{i=1}^{N}|S_i - \widehat{S}_i|^2}{\sum_{i=1}^{N}|S_i|^2}} \times 100 \qquad \text{Eq. (6.23)}$$

where S_i is the ideal constellation point, and $\widehat{S}_i$ is the corresponding symbol generated by the neural network (see Figure 6-15). Lower EVM values indicate higher fidelity in reproducing the target constellation, and therefore better encoder performance.

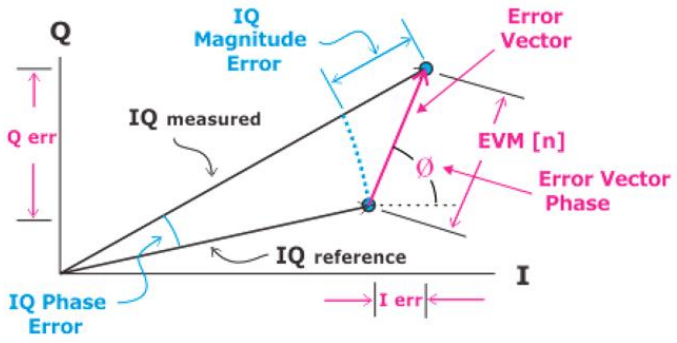


Figure 6-15 EVM Illustration [71]

This experimental setup enables a rigorous, data-driven training process that is grounded in standards-compliant modulation behavior. Moreover, by evaluating the encoder's output in terms of EVM for each modulation scheme, we are able to benchmark the effectiveness of the learned mappings and provide a fair comparison between different dynamic strategies based on accuracy, complexity, and adaptability. Figure 6-16 shows the ideal constellation points and the DL-based generated constellation for QPSK.

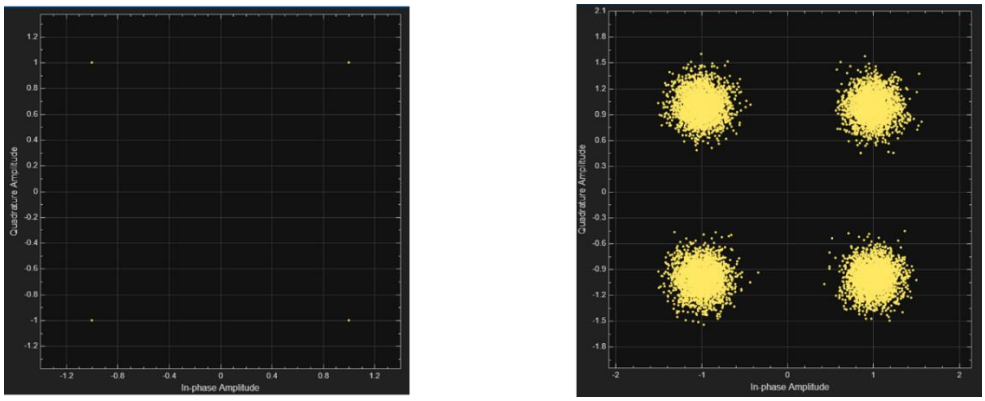


Figure 6-16 Ideal and DL-based generated Constellation Points for QPSK

EVM measurements shown in Figure 6-17 confirm that the DL-based generated constellation points achieve $EVM_{RMS} = 9.01\%$, which is 27.9% less than the 12.5 %EVM value required by standard IEEE 802.11n standard for QPSK modulation [71].

EVM / MER	Values
RMS EVM (%)	9.01
Peak EVM (%)	29.58
Avg EVM (dB)	-20.91
Peak EVM (dB)	-10.58
Avg MER (dB)	20.91

Figure 6-17 EVM Values for DL-Based QPSK Encoder

Figure 6-18 shows the convergence of EVM value with the training epochs, where each epoch corresponds to 20 training iterations.

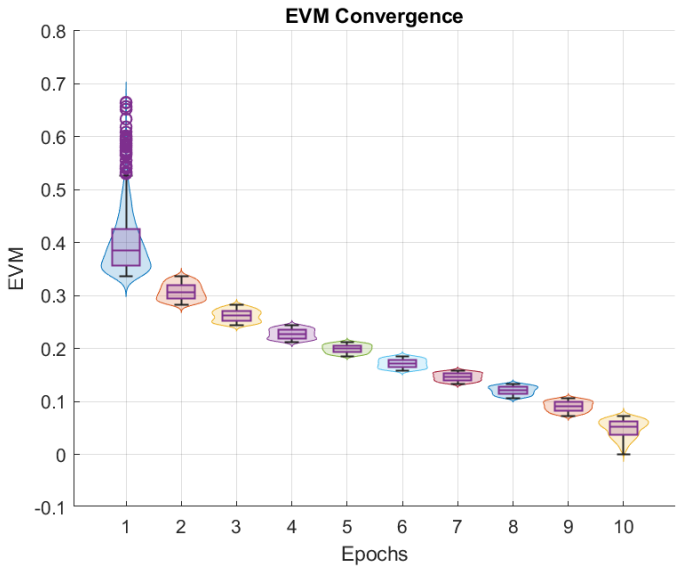
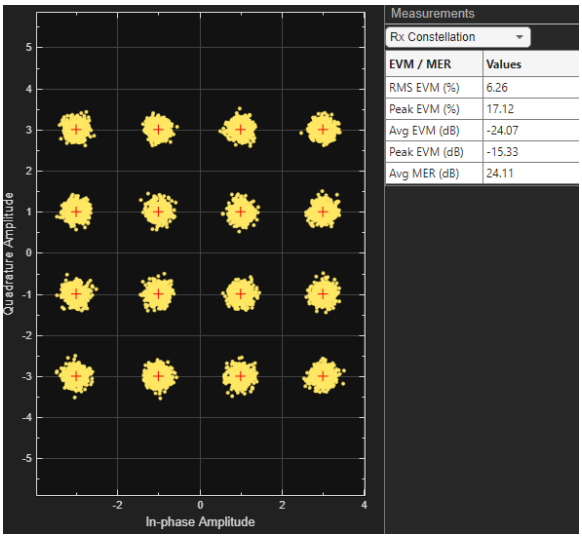
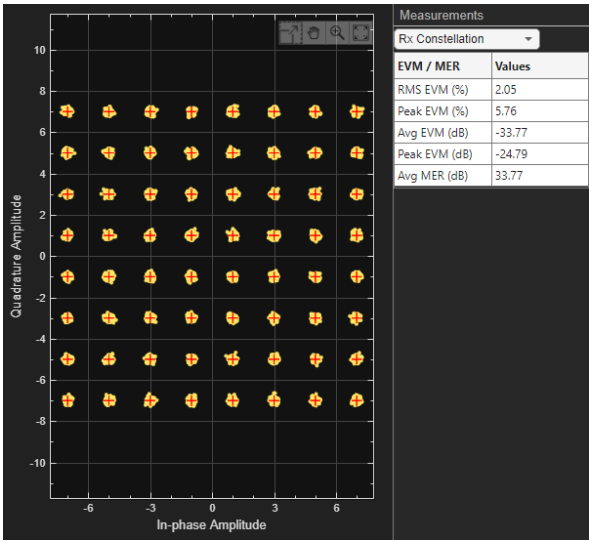


Figure 6-18 EVM Convergence Vs. Training Epochs

Similarly, EVM results confirm that DL-based encoders for 16-QAM and 64-QAM outperform the transmitter requirements [71] for generated constellation points by 29.4% and 48.75% respectively (see Figure 6-19 a and b).



(a)



(b)

Figure 6-19 EVM Values (a) DL-Based 16-QAM encoder (b) DL-based 64-QAM Encoder

6.5 Adaptive Neural Transceiver Design: Integrating Dynamic Encoder and Decoder in a Practical Autoencoder Framework

Having independently explored the design, implementation, and evaluation of dynamic neural decoders and encoders tailored for adaptive modulation in IEEE 802.11n, this section presents the natural progression toward a **joint end-to-end deep learning-based communication system**. The goal of this system is to integrate the **modulation-aware dynamic encoder and decoder** into a single, trainable autoencoder architecture that learns to optimize the entire wireless link—from bitstream to decoded output—under varying modulation constraints.

In contrast to conventional systems, where the transmitter and receiver are designed as separate blocks using fixed signal processing pipelines, the proposed architecture models both components as neural networks that can **co-adapt through joint training**. This approach enables end-to-end optimization, allowing the encoder and decoder to learn how to best work together under real-world channel conditions, hardware nonlinearities, and modulation dynamics.

To bridge the gap between simulation and practical deployment, the proposed system is implemented and evaluated using a **Software Defined Radio (SDR)-based testbed**. Specifically, a pair of **NI USRP X310 SDRs** are used to transmit and receive IEEE 802.11n-compliant OFDM signals over the air.

The **dynamic neural encoder** generates modulation-specific constellation points, which are transmitted through the USRP, while the **dynamic neural decoder** receives the over-the-air waveform and reconstructs the original bitstream. The use of real-time wireless transmission enables the system to capture realistic channel impairments, including RF front-end nonidealities, noise, and frequency offset—thereby validating the performance of the autoencoder under operational conditions.

In addition to real-world implementation, the system incorporates **modulation-aware dynamic structures** in both the encoder and decoder, enabling it to adapt its computational complexity and inference path based on the modulation scheme (QPSK, 16-QAM, or 64-QAM). This modulation adaptivity provides the basis for efficient resource usage and robustness across varying channel conditions. Training is conducted using a fine-tuning approach, leveraging

pretrained encoder and decoder weights and optimizing them jointly with real-world channel traces.

6.5.1 System Overview

The proposed end-to-end deep learning-based communication system integrates the dynamic neural encoder and decoder within a unified **autoencoder framework** designed for modulation-adaptive communication over IEEE 802.11n. The architecture supports real-time adaptation to modulation orders based on channel conditions, optimizing both computational efficiency and decoding robustness.

6.5.1.1 Dynamic Neural Encoder

As detailed earlier, the encoder is implemented using a **modulation network switching strategy**, comprising a single architecture with three parallel subnetworks. Each subnetwork is independently trained to map binary input sequences to constellation symbols for a specific modulation order—**QPSK, 16-QAM, or 64-QAM**. We selected this structure for the encoder due to its energy efficiency (see table 6-1).

Unlike static encoders, where a fixed modulation is assumed, this encoder dynamically selects the appropriate branch at runtime based on channel conditions. Specifically, **conventional channel state information (CSI)** is used to estimate the current signal-to-noise ratio (SNR), and based on IEEE 802.11n SNR thresholds, the encoder selects the modulation order accordingly. For instance:

- QPSK is used when SNR ≤ 12 dB
- 16-QAM is used for 12 dB < SNR ≤ 18 dB
- 64-QAM is used for SNR > 18 dB

This CSI-driven control logic ensures that the encoder adjusts its structure proactively, selecting a lower-complexity modulation branch in poor channel conditions to maximize reliability, and switching to higher-order modulations when the channel permits, improving spectral efficiency. The switching mechanism is embedded in the encoder runtime, enabling fast and deterministic modulation adaptation.

6.5.1.2 Dynamic Neural Decoder

As described, the decoder is designed to operate using one of three dynamic structures, each activated based on the modulation detected from the received signal:

- **Early Exit Architecture** allows inference to terminate at intermediate layers when the decoder is sufficiently confident, reducing complexity for low-order modulations.
- **Modulation-Aware Network Switching** uses dedicated decoders for each modulation order, selected via a learned modulation classifier.
- **Dynamic Weight Reloading** retains a single decoder structure and switches parameter sets according to the detected modulation.

These decoder architectures complement the encoder's modulation-specific outputs, enabling the full system to dynamically optimize both transmitter and receiver paths based on SNR feedback and real-time link conditions.

Together, these components form a modulation-adaptive deep learning-based wireless autoencoder, where conventional CSI is used to control encoder switching, and modulation classification guides decoder adaptation. This framework enables efficient, scalable, and robust communication across a wide range of channel conditions, while offering meaningful reductions in energy consumption and computation.

6.5.2 Fine-Tuning Strategy

Rather than training the end-to-end autoencoder system from scratch, a **fine-tuning approach** was employed to maximize the benefits of previously trained modular components while reducing overall training complexity and convergence time. The dynamic neural encoder and decoder architectures, each originally trained independently on their respective tasks—symbol generation and bit recovery—were first integrated into the joint autoencoder framework. During initial integration, the pretrained weights of the encoder and decoder subnetworks for each modulation order (QPSK, 16-QAM, 64-QAM) were **frozen** to preserve their modulation-specific capabilities.

Subsequently, **fine-tuning was performed selectively**, where either the encoder or decoder (or both) were unfrozen and updated during joint training, depending on the modulation type and channel conditions. This selective adaptation allowed the system to co-optimize symbol mapping and recovery without losing the robustness achieved during isolated training.

Let the encoder and decoder be modeled as parameterized functions:

$$z = f_{\theta_E}(x) \quad \text{and} \quad \hat{x} = g_{\theta_D}(y)$$

where:

- $x \in \{0,1\}^k$ is input bitstream.
- $z \in \mathbb{C}^n$ is the encoded symbol vector.
- $y \in \mathbb{C}_{\mathcal{N}}^n$ is the received symbol vector after channel distortion.
- $\hat{x} \in \{0,1\}^k$ is the reconstructed bitstream.
- f_{θ_E} and g_{θ_D} represent the encoder and decoder networks.

6.5.2.1 Independent Pretraining

During pretraining, the encoder and decoder are optimized separately using task-specific loss functions. For the encoder (e.g., constellation learning), the objective is:

$$\min_{\theta} \mathcal{L}_{enc} = \frac{1}{N}\sum_{i=1}^{N} \left\| f_{\theta_E}(x_i) - z_i^{target} \right\|^2 \qquad \text{Eq. (6.24)}$$

For the decoder, the supervised reconstruction loss (e.g., cross-entropy) is:

$$\min_{\theta} \mathcal{L}_{dec} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{H}\left(\hat{x}_i, g_{\theta_D}(y_i) \right) \qquad \text{Eq. (6.25)}$$

where $\mathcal{H}(\,.\,,.\,)$ is the cross entropy loss between the ground truth and the decoder output.

6.5.2.2 End-to-End Fine-Tuning

After pretraining, the encoder and decoder are integrated into a joint autoencoder system, and a **fine-tuning phase** is initiated. Instead of training the entire model from scratch, the fine-tuning process begins with the pretrained weights $\theta_E^{(0)}, \theta_D^{(0)}$ and selectively updates them to minimize end-to-end loss:

$$\min_{\theta_E^{(0)},\theta_D^{(0)}} \mathcal{L}_{AE} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{H}\left(\hat{x}_i, g_{\theta_D}\left(h\left(f_{\theta_E}(x)\right)\right)\right) \quad \text{Eq. (6.26)}$$

where $h(.)$ Denotes the channel impairments affecting the encoded symbols.

This is equivalent to applying a **parameter update rule** only on a subset of the model:

$$\theta^{(t+1)} = \theta^{(t)} - \eta.\nabla_\theta \mathcal{L}_{AE}(\theta) \quad \text{Eq. (6.27)}$$

with $\theta \in \{\theta_E, \theta_D\}$.

η is the learning rate. Fine-tuning uses lower learning rates than pretraining to ensure stability and preserve pretrained knowledge.

The use of fine-tuning, rather than full retraining, offered several key advantages:

- **Preserved Modulation Fidelity:** The encoder retained its ability to produce constellation-accurate outputs already learned from high-quality signal generator references.
- **Reduced Training Time:** Fine-tuning converged significantly faster than full end-to-end retraining, as most of the parameter space was already well-initialized.
- **Improved Generalization:** Starting from pretrained weights allowed the model to generalize better across modulation orders and SNR regimes, particularly in lower-SNR conditions where random initialization may lead to unstable learning.

Overall, this fine-tuning strategy ensured that the dynamic end-to-end autoencoder system remained modular, interpretable, and resource-efficient, while still achieving joint performance gains through selective reoptimization.

6.6 Results and Discussion:

This section depicts the results of BLER for the joint end-to-end dynamci autoencoder based wireless communication system. Figure 6-20 shows the structure of the dynamic wireless AE implemented using USRP X310 for network switching decoder.

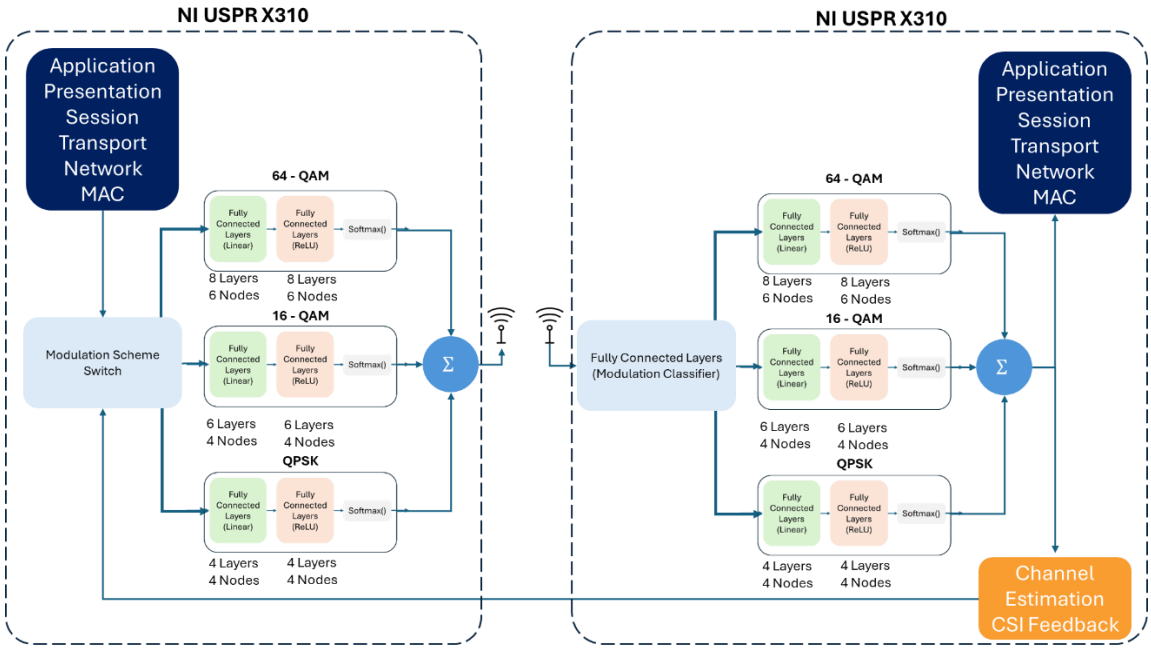


Figure 6-20 End-to-End Dynamic Wireless SDR-Autoencoder (Network Switching Decoder)

Figure 6-21 presents the block error rate (BLER) performance of the proposed network selection-based dynamic neural decoder compared to conventional receivers for three modulation schemes: QPSK, 16-QAM, and 64-QAM, over a range of E_b/N_o values from 0 to 8 dB. The results clearly demonstrate the advantage of the dynamic AE across all modulation orders.

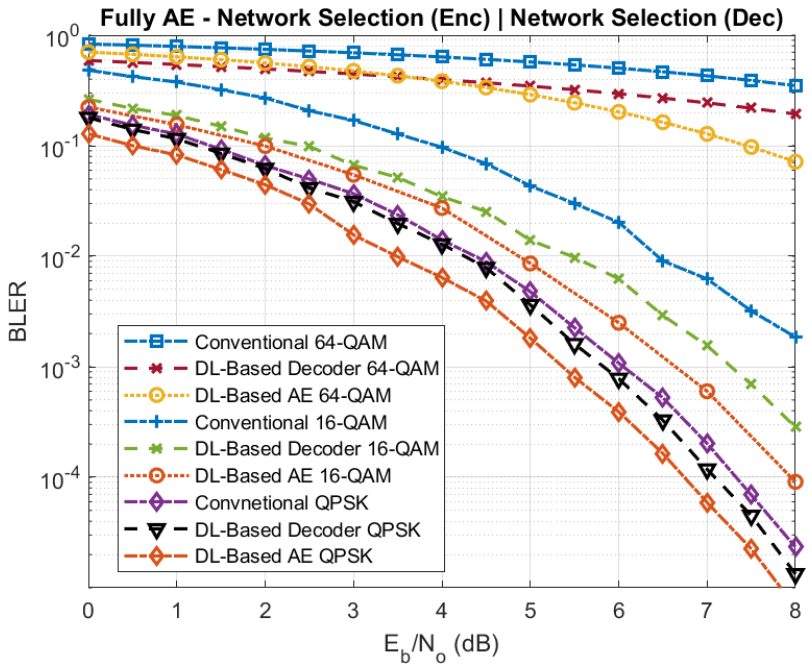


Figure 6-21 BLER Performance for Wireless AE (Network Switching Decoder)

For QPSK, the wireless AE consistently outperforms the conventional transceiver, achieving more than 1 dB gain at a BLER of 10^{-2}. A similar trend is observed for 16-QAM, where the DL-based system achieves notable improvements in the moderate SNR regime (2–6 dB), reducing BLER by nearly an order of magnitude compared to its conventional counterpart. The 64-QAM case shows the smallest relative gain due to the increased constellation complexity and the higher sensitivity to noise; however, the DL-based decoder still provides a consistent performance improvement, particularly at higher E_b/N_o values.

Overall, the results validate the effectiveness of modulation-aware neural network switching, where dedicated decoders trained per modulation scheme can achieve superior performance in both low and moderate SNR regimes. The performance gap between the DL-based and conventional systems increases as the modulation order decreases, indicating that neural decoders benefit from the simpler symbol structure in lower-order constellations. These findings highlight the potential of dynamic deep learning-based decoding in delivering high accuracy while maintaining adaptability and scalability across varying modulation configurations.

The structure of the second AE type is illustrated in Figure 6-22, in which the decoder is realized using parameter adaption approach.

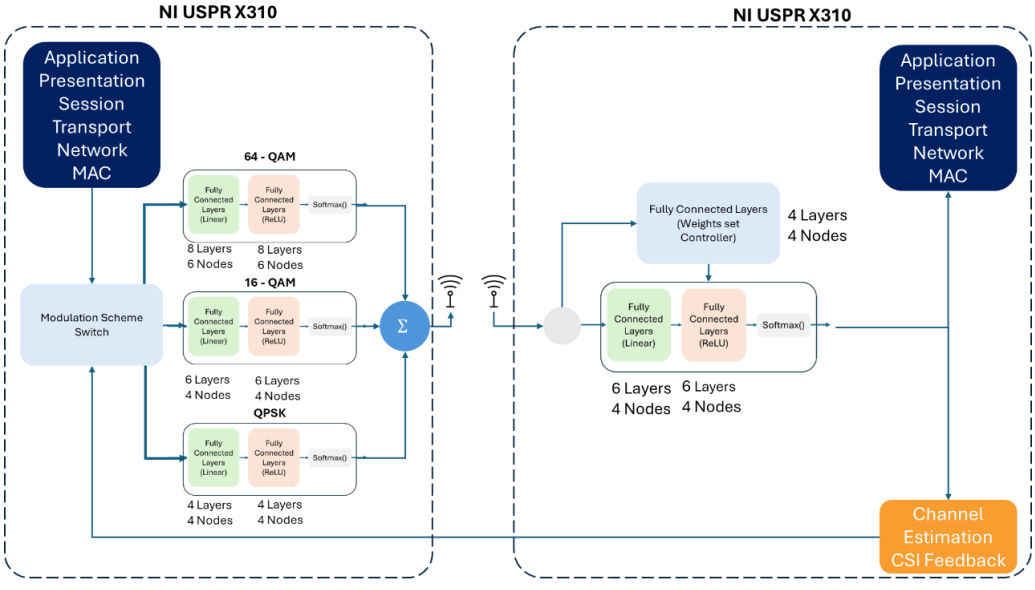


Figure 6-22 End-to-End Dynamic Wireless SDR-Autoencoder (Parameter Adaption Decoder)

Figure 6-23 illustrates BLER performance of the dynamic AE implemented with weight reloading decoder, where a single decoder architecture is reused across all modulation schemes (QPSK, 16-QAM, and 64-QAM) with separate parameter sets loaded dynamically based on the modulation order. This is compared to conventional decoding methods under identical conditions and modulation configurations.

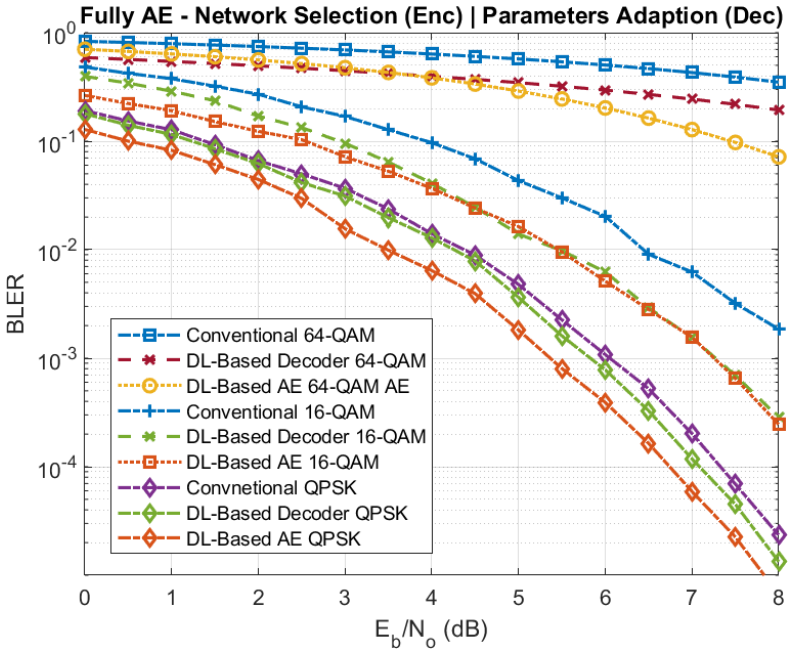


Figure 6-23 BLER Performance for Wireless AE (Parameter Adaption Decoder)

The results show a consistent performance improvement of the DL-based decoder with parameter adaptation across all modulation schemes. For **QPSK**, the DL-based decoder achieves the lowest BLER curve, maintaining nearly a full order-of-magnitude improvement over the conventional decoder across the entire E_b/N_o range. Similar gains are observed for **16-QAM**, where the dynamic decoder exhibits superior robustness, particularly in the mid-SNR regime (3–6 dB), achieving over E_b/N_oat a BLER of 10^{-2}. While the performance improvement is slightly narrower for **64-QAM**, the DL-based decoder still shows measurable gains, especially at higher E_b/N_o values, where it benefits from better parameter fine-tuning and less constellation overlap.

These findings validate the effectiveness of the **dynamic weight reloading strategy**, which combines the flexibility of a shared architecture with the specialization of modulation-specific parameter sets. This approach provides a scalable and memory-efficient design, avoiding the need to maintain multiple decoder structures while still delivering performance comparable to

fully dedicated models. The parameter adaptation mechanism proves particularly beneficial for low and mid-order modulations, which benefit from lightweight yet optimized configurations, ultimately contributing to an overall reduction in computational cost without sacrificing accuracy.

The last structure displayed in Figure 6-24 represents the realization of fully DL-based wireless system using early exit based decoder. In the early-exit architecture, a single deep decoder network is trained with multiple intermediate classifiers, allowing the inference process to terminate earlier for simpler modulation schemes such as QPSK and 16-QAM. This structure enables computational savings by skipping deeper layers when not required, while still processing higher-order modulations such as 64-QAM through the full network depth.

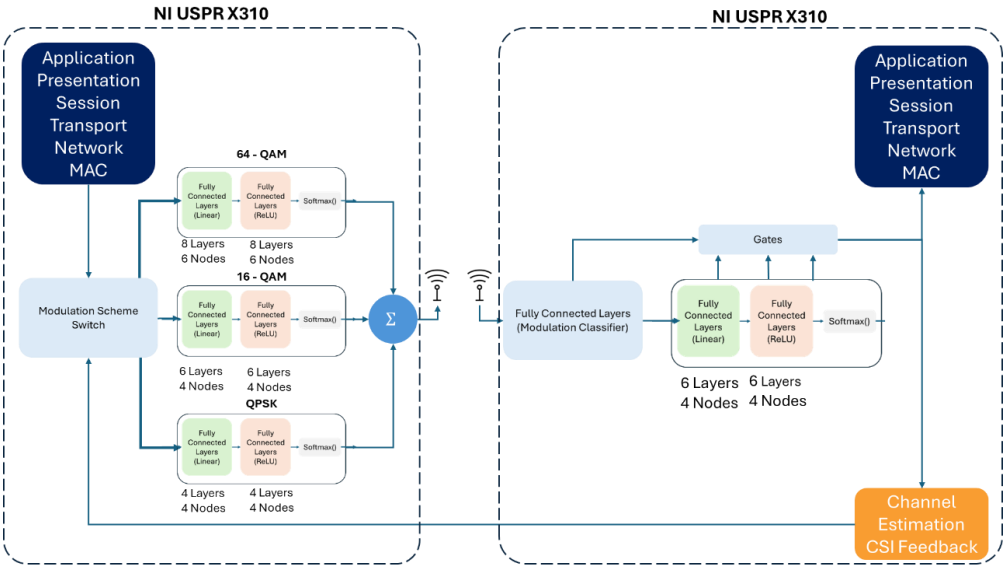


Figure 6-24 End-to-End Dynamic Wireless SDR-Autoencoder (Early Exit Decoder)

The results (illustrated in Figure 6-25) demonstrate that this structure achieves comparable or improved BLER performance relative to conventional decoders across all modulation schemes. For QPSK, the early-exit mechanism allows the network to make confident predictions with only shallow layers, achieving a performance closely matching the conventional decoder, and even outperforming it slightly at moderate-to-high SNR. The performance for 16-QAM shows a clear advantage in the mid-SNR range, with the DL-based decoder achieving more than 1 dB gain at a BLER of 10^{-2}. For 64-QAM, where deeper network layers are fully activated due to higher complexity, the DL-based decoder continues to show improved robustness, especially beyond 5 dB, reducing the error floor compared to the conventional baseline.

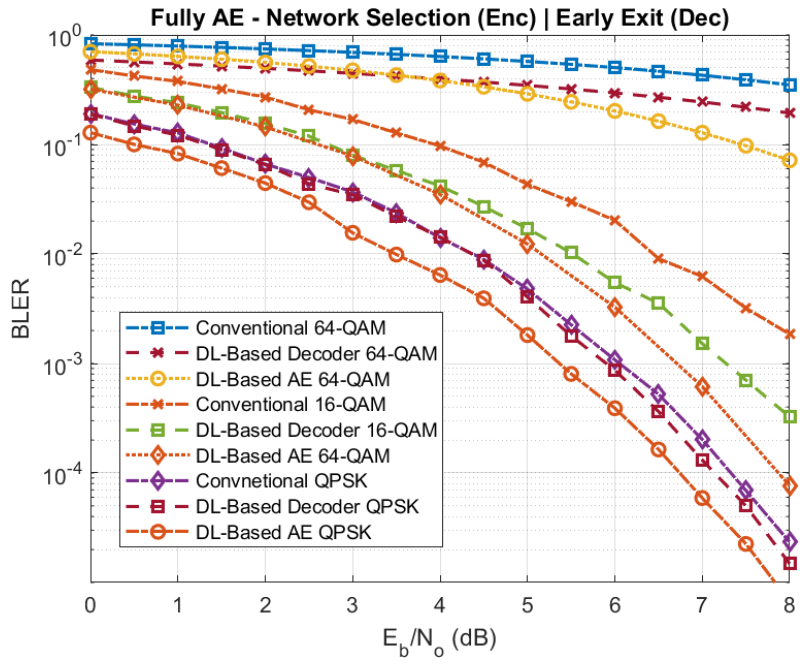


Figure 6-25 BLER Performance for Wireless AE (Early Exit Decoder)

These findings confirm that the early-exit strategy not only reduces unnecessary computation for low-complexity inputs but also maintains high decoding accuracy across all modulation orders. The design is especially beneficial for energy-efficient inference, as it leverages the correlation between modulation order and decoding complexity to dynamically adjust the computational path. This aligns well with the requirements of adaptive modulation systems such as IEEE 802.11n, where different modulation schemes are selected in real-time based on channel quality.

6.6.1 BLER Performance Comparison

Across all three strategies, the DL-based decoders consistently outperform their conventional counterparts for each modulation scheme. However, the degree and nature of performance improvement vary:

- **Network Selection:**
    - Provides modest but stable gains across all modulation schemes.
    - Offers ~1 dB SNR gain at BLER =10^{-2} for 16-QAM and QPSK.
    - Effective due to dedicated networks optimized for each modulation, but incurs higher memory usage due to the need for three full networks.
- **Parameter Adaptation / Weight Reloading:**

- Delivers the **best overall performance**, particularly for 16-QAM and QPSK, achieving up to **2 dB gain** over conventional decoders.
- Benefits from weight specialization while sharing a single decoder structure, balancing performance and memory efficiency.
- Slightly lower performance improvement for 64-QAM, where decoding complexity may challenge generalization.

- **Early Exit:**
  - Offers competitive performance with reduced computation, especially for QPSK and 16-QAM.
  - For QPSK, performance nearly matches network selection and weight reloading but with fewer executed layers.
  - Slightly lower improvement for 64-QAM due to full network execution and absence of structure specialization.

Table 6-2 Efficiency and Architectural Trade-offs

Strategy	Modularity	Memory Cost	Computation Adaptivity	BLER Improvement	Strengths
Network Selection	3 independent networks	High	Modulation-level only	Moderate	Strong specialization
Weight Reloading	1 network × 3 sets of weights	Medium	Modulation-level only	High	Best performance–efficiency trade-off
Early Exit	1 network	Low	Fine-grained (layer-wise)	Good	Best for energy-constrained devices

6.6.2 Modulation-Specific Observations

- **QPSK:**

- All three DL-based strategies achieve large gains due to lower constellation complexity.
- Early Exit performs especially well due to shallow path activation.

- **16-QAM:**
  - Weight Reloading outperforms others, balancing complexity and representational depth.
- **64-QAM:**
  - Gains are more modest across all methods.
  - Network Selection and Weight Reloading perform slightly better due to higher specialization.
  - Early Exit still maintains gains but lacks structural adaptation beyond full depth.

7 Conclusion, Open Research Challenges

This work explored the integration of deep learning into wireless communication systems, focusing on the design and evaluation of dynamic neural network architectures that can adapt in real time to varying modulation schemes, channel conditions, and computational constraints. The motivation behind this work stemmed from the growing complexity and performance demands of modern wireless networks, which traditional block-based communication systemsrelying on fixed signal processing algorithms struggle to meet effectively. To address these challenges, this research proposed a modular, end-to-end learning framework based on dynamic neural networks that can intelligently adjust their internal structure and computation paths during inference.

The core idea of this work was to leverage deep learning not only to replace individual components such as demodulators or equalizers but to reimagine the entire transceiver as a trainable autoencoder that adapts to the environment. A dynamic neural encoder was developed to learn how to map bitstreams into modulation-specific constellation points, adapting to QPSK, 16-QAM, or 64-QAM based on real-time channel state information (CSI). On the receiver side, three dynamic decoding strategies modulation-aware network switching, dynamic weight reloading, and early exit inferencewere implemented and compared to static conventional decoders. These strategies allowed the decoder to scale its complexity depending on the modulation order or the confidence level of intermediate layers, resulting in significant reductions in energy consumption and floating-point operations (FLOPs) without compromising performance.

A complete end-to-end deep learning-based autoencoder system was implemented by integrating the encoder and decoder modules and fine-tuning them jointly over realistic wireless channels. Rather than retraining the entire model from scratch, a fine-tuning approach was adopted to preserve the strengths of pretrained subnetworks while optimizing the full pipeline. The system was evaluated under both simulated and real-world conditions using a software-defined radio (SDR) platform built on USRP X310 and Signal Hound VSG60A hardware. The collected over-the-air data captured realistic channel impairments, including RF nonlinearities, fading, and noise. Performance was quantified in terms of block error rate (BLER), error vector

magnitude (EVM), computational complexity, and energy efficiency. Across all metrics, the proposed dynamic architectures consistently outperformed conventional IEEE 802.11n receivers, particularly in low-to-mid SNR regimes.

This research demonstrated that dynamic neural architectures are highly effective in adapting communication systems to fluctuating conditions and requirements. The encoder and decoder co-adapt to each other and to the channel, enabling performance gains that would not be possible with independently designed or static models. More importantly, the ability to dynamically switch structures or parameters or exit computation early makes these architectures especially suitable for energy-constrained or latency-sensitive applications. The work also highlighted key implementation challenges, such as the need to balance memory and speed in network switching, and the importance of robust modulation classification to guide structural adaptation.

The key contributions of this dissertation span across decoder design, encoder modeling, end-to-end joint optimization, and hardware-in-the-loop experimental validation:

1. **Dynamic Neural Decoder Architectures:**
    - Three distinct dynamic strategies were proposed: **modulation-aware network switching**, **parameter reloading**, and **early exit inference**.
    - Each decoder was tailored to handle QPSK, 16-QAM, and 64-QAM constellations using a modular, resource-efficient design.
    - These dynamic structures were experimentally shown to reduce **floating-point operations (FLOPs)** and energy consumption, while achieving superior **block error rate (BLER)** performance compared to conventional OFDM receivers.
2. **Dynamic Neural Encoder Design:**
    - A modulation-aware switching encoder was developed to generate constellation points for each modulation scheme using a unified architecture.

- Encoder switching was controlled by **conventional channel state information (CSI)**, enabling real-time adaptability based on SNR thresholds.
- The encoder was trained using real IEEE 802.11n signals generated by a Signal Hound VSG60A, and evaluated using **error vector magnitude (EVM)**.

3. **Joint End-to-End Autoencoder System:**
   - A complete deep learning-based autoencoder framework was implemented, integrating the dynamic encoder and decoder in a single pipeline.
   - Fine-tuning was employed instead of retraining, leveraging pretrained models and accelerating convergence while retaining modulation-specific knowledge.
   - The autoencoder was deployed on a **USRP X310-based SDR testbed**, allowing for over-the-air evaluation of the system under realistic indoor multipath conditions.
4. **Experimental Validation and Analysis:**
   - The system was benchmarked on BLER, FLOPs, EVM, and energy consumption across multiple dynamic strategies.
   - Results demonstrated that the **dynamic decoder with parameter reloading** achieved the best BLER performance, while **early exit decoding** provided the most efficient inference path for low-order modulations.
   - The joint system outperformed modular baselines and conventional IEEE 802.11n receivers, validating the value of co-optimized transceiver design.

The implications of this research extend well beyond the boundaries of IEEE 802.11n. The concept of **dynamic neural networks** has broad applicability in future wireless systems such as **5G-Advanced, 6G, and beyond**, where adaptability, scalability, and low-latency operation are critical. Dynamic computation allows systems to tailor resource usage based on data complexity, modulation order, channel conditions, or latency requirements. This paradigm opens up new frontiers in the design of:

- **Cognitive Radios** that adapt processing depth based on interference levels.
- **Low-power IoT Devices** that utilize early exits to conserve energy.
- **Massive MIMO Systems** that switch decoder complexity based on spatial stream separability.
- **Secure and Robust Communications** using dynamic defense mechanisms against adversarial interference.

Furthermore, the joint encoder-decoder co-design offers a blueprint for **end-to-end learning in communication systems**, shifting the design focus from handcrafted modules to learned, data-driven representations. This enables transceivers to co-optimize for channel impairments, hardware limitations, and power constraints in ways that static systems cannot.

7.1 Limitations and Challenges

Despite its promise, the deployment of dynamic neural architectures in wireless systems comes with a set of practical challenges:

- **Latency vs. Accuracy Trade-offs:** Early exit and weight reloading strategies require careful balancing between computational savings and decoding fidelity.
- **Model Management Overhead:** Maintaining multiple parameter sets or networks introduces memory and switching overhead, which may be constrained in embedded systems.
- **Hardware Deployment:** Integration of dynamic models in FPGA or ASIC platforms requires efficient support for branching and conditional computation.
- **Generalization Across Environments:** While tested in real-world indoor conditions, further evaluation is needed in outdoor, mobile, and high-speed scenarios.

Addressing these challenges will require new algorithms for dynamic control, neural architecture compression, hardware-aware training, and adaptive optimization.

7.2 Open Research Topics

This work opens several promising directions for future research:

1. **Cross-Layer Adaptivity:** Leveraging PHY-level dynamicity to inform MAC and network-layer decisions such as retransmission, scheduling, and beam selection.
2. **Online Learning:** Implementing online adaptation mechanisms that fine-tune encoder or decoder weights during runtime based on feedback.
3. **Multi-User and Interference-Aware Systems:** Expanding the dynamic autoencoder to support interference cancellation, multi-user MIMO, and NOMA scenarios.
4. **Hardware Acceleration:** Deploying dynamic inference paths on edge AI chips or FPGAs to study power-latency trade-offs in real time.

7.3 Final Remarks

In conclusion, this dissertation demonstrated that **deep learning and dynamic NN** offer a transformative approach to designing adaptive, efficient, and high-performance wireless communication systems. By rethinking the transceiver as a flexible neural architecture rather than a fixed signal chain, we unlock new levels of agility and intelligence. The proposed dynamic encoder-decoder system represents a step forward toward intelligent, context-aware, and resource-optimized wireless PHY layers that are ready to meet the challenges of next-generation connectivity.

References

[1]. W. Yu, F. Sohrabi and T. Jiang, "Role of Deep Learning in Wireless Communications," in *IEEE BITS the Information Theory Magazine*, vol. 2, no. 2, pp. 56-72, 1 Nov. 2022, doi: 10.1109/MBITS.2022.3212978.

[2]. M. Di Renzo et al., "Smart radio environments empowered by reconfigurable intelligent surfaces: How it works, state of research, and road ahead," IEEE J. Sel. Areas Commun., vol. 38, no. 11, pp. 2450–2525, Jul. 2019

[3]. Mousavi and R. G. Baraniuk, "Learning to invert: Signal recovery via Deep Convolutional Networks," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, 2017, pp. 2272-2276,

[4]. Z. Qin, H. Ye, G. Y. Li and B. -H. F. Juang, "Deep Learning in Physical Layer Communications," in *IEEE Wireless Communications*, vol. 26, no. 2, pp. 93-99, April 2019, doi: 10.1109/MWC.2019.1800601.

[5]. C. A. Metzler, A. Maleki and R. G. Baraniuk, "From Denoising to Compressed Sensing," in *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5117-5144, Sept. 2016

[6]. Metzler, Chris, Ali Mousavi, and Richard Baraniuk. "Learned D-AMP: Principled neural network based compressive image recovery." *Advances in neural information processing systems* 30 (2017).

[7]. C. -K. Wen, W. -T. Shih and S. Jin, "Deep Learning for Massive MIMO CSI Feedback," in *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748-751, Oct. 2018,

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

[9]. X. Li *et al.*, "A survey on deep learning techniques in wireless signal recognition," *Wireless Communications and Mobile Computing*, vol. 2019, Art. no. 5629572, 2019.

[10]. J. Chen, Y. Gao, Y. Zhou, Z. Liu, D. p. Li and M. Zhang, "Machine Learning enabled Wireless Communication Network System," *2022 International Wireless Communications and Mobile Computing (IWCMC)*, Dubrovnik, Croatia, 2022, pp. 1285-1289

[11]. T. En, W. Ye, D. Fei, P. Zhiwen and Y. Xiaohu, "A practical eNB off/on based energy saving scheme for real LTE networks," *2015 17th International Conference on Advanced Communication Technology (ICACT)*, PyeongChang, Korea (South), 2015, pp. 12-17

[12]. T. J. O'Shea, T. Roy, N. West and B. C. Hilburn, "Physical Layer Communications System Design Over-the-Air Using Adversarial Networks," *2018 26th European Signal Processing Conference (EUSIPCO)*, Rome, Italy, 2018, pp. 529-532

[13]. A. Archi, H. Ait Saadi, and S. Mekaoui, "Applications of deep reinforcement learning in wireless networks—A recent review," in Proc. 2023 2nd Int. Conf. Electronics, Energy and Measurement (IC2EM), vol. 1, 2023, pp. 1–6,

[14]. Erpek, Tugba, et al. "Deep learning for wireless communications." Development and Analysis of Deep Learning Architectures (2020): 223-266.

[15]. T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563-575

[16]. "What is an autoencoder?," IBM, https://www.ibm.com/topics/autoencoder (accessed Feb. 21, 2024).

[17]. Vanhoucke, Vincent, Andrew Senior, and Mark Z. Mao. "Improving the speed of neural networks on CPUs." Proc. deep learning and unsupervised feature learning NIPS workshop. Vol. 1. No. 2011. 2011.

[18]. Davaslioglu, Kemal, Tugba Erpek, and Yalin E. Sagduyu. "End-to-end autoencoder communications with optimized interference suppression." arXiv preprint arXiv:2201.01388 (2021).

[19]. C. Zou, F. Yang, J. Song and Z. Han, "Channel Autoencoder for Wireless Communication: State of the Art, Challenges, and Trends," in *IEEE Communications Magazine*, vol. 59, no. 5, pp. 136-142, May 2021
[20]. P. Baldi, “Autoencoders,” in Deep Learning in Science, Cambridge: Cambridge University Press, 2021, pp. 71–98
[21]. M. Tran, “Understanding U-net,” Medium, https://towardsdatascience.com/understanding-u-net-61276b10f360 (accessed Feb. 26, 2024).
[22]. S. Bishayee, “The basic concept of Autoencoder-the self-supervised deep learning,” Medium, https://medium.com/@soumallya160/the-basic-concept-of-autoencoder-the-self-supervised-deep-learning-454e75d93a04 (accessed Jan. 24, 2024).
[23]. Yu, Shujian, and Jose C. Principe. "Understanding autoencoders with information theoretic concepts." Neural Networks 117 (2019): 104-123.
[24]. E. Ordway-West, P. Parveen and A. Henslee, "Autoencoder Evaluation and Hyper-Parameter Tuning in an Unsupervised Setting," *2018 IEEE International Congress on Big Data (BigData Congress)*, San Francisco, CA, USA, 2018, pp. 205-209
[25]. “Autoencoders in Deep learning: Tutorial & use cases [2023],” V7, https://www.v7labs.com/blog/autoencoders-guide#h4 (accessed Jan. 13, 2024).
[26]. M. D. Siddiqi, B. Jiang, R. Asadi, and A. Regan, “Hyperparameter tuning to optimize implementations of denoising autoencoders for imputation of missing spatio-temporal data,” Procedia Computer Science, vol. 184, pp. 107–114, 2021.
[27]. Mechelli and S. Vieira, in Machine learning: Methods and applications to brain disorders, London: Elsevier Academic Press, 2020.
[28]. Goodfellow, Y. Bengio, and A. Courville, “Autoencoders,” in Deep learning -- Grundlagen, Aktuelle Verfahren und Algorithmen, Neue Forschungsansätze, mitp Verlag, 2018
[29]. Jeremy Jordan, “Introduction to autoencoders.,” Jeremy Jordan, https://www.jeremyjordan.me/autoencoders/ (accessed Mar. 29, 2024).
[30]. Felix, S. Cammerer, S. Dörner, J. Hoydis and S. Ten Brink, "OFDM-Autoencoder for End-to-End Learning of Communications Systems," *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Kalamata, Greece, 2018, pp. 1-5
[31]. Y. An, S. Wang, L. Zhao, Z. Ji and I. Ganchev, "A Learning-Based End-to-End Wireless Communication System Utilizing a Deep Neural Network Channel Module," in *IEEE Access*, vol. 11, pp. 17441-17453, 2023
[32]. V. Raj and S. Kalyani, "Design of Communication Systems Using Deep Learning: A Variational Inference Perspective," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1320-1334, Dec. 2020
[33]. H. Ye, L. Liang, G. Y. Li and B. -H. Juang, "Deep Learning-Based End-to-End Wireless Communication Systems With Conditional GANs as Unknown Channels," in *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3133-3143
[34]. V. Raj and S. Kalyani, "Backpropagating Through the Air: Deep Learning at Physical Layer Without Channel Models," in *IEEE Communications Letters*, vol. 22, no. 11, pp. 2278-2281, Nov. 2018
[35]. F. A. Aoudia and J. Hoydis, "Model-Free Training of End-to-End Communication Systems," in *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 11, pp. 2503-2516, Nov. 2019
[36]. Shafi, A. Kumar, and R. Nakkeeran, "Performance analysis of free space optical networks under external limiting factors," Results in Optics, vol. 14, p. 100615, 2024.
[37]. I. Abdalla, M. B. Rahaim, and T. D. C. Little, "Interference in multi-user optical wireless communications systems," Philos. Trans. R. Soc. A Math. Phys. Eng. Sci., Apr. 2020.
[38]. M. Asim, M. K. Singh, K. N. Singh, L. M., and R. Rani, "Application of deep learning in free space optical communication using Malaga channel," in Proc. 2023 3rd Int. Conf. Adv. Electron. Commun. Eng. (AECE), Ghaziabad, India, 2023, pp. 378–382.

[39]. M. A. Amirabadi et al., "A survey on machine and deep learning for optical communications," arXiv preprint arXiv:2412.17826, 2024.
[40]. M. Soltani, W. Fatnassi, A. Aboutaleb, Z. Rezki, A. Bhuyan, and P. Titus, "Autoencoder-Based Optical Wireless Communications Systems," in Proc. IEEE Globecom Workshops (GC Wkshps), Abu Dhabi, United Arab Emirates, 2018, pp. 1–6, doi: 10.1109/GLOCOMW.2018.8644104.
[41]. T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," IEEE Trans. Cogn. Commun. Netw., vol. 3, no. 4, pp. 563–575, Dec. 2017.
[42]. X. Liu, Z. Wei, A. Pepe, Z. Wang, and H. Y. Fu, "Autoencoder for Optical Wireless Communication System in Atmospheric Turbulence," in Proc. 2020 Opto-Electron. Commun. Conf. (OECC), Taipei, Taiwan, 2020, pp. 1–3.
[43]. M. A. Amirabadi, M. H. Kahaei, and S. A. Nezamalhosseni, "Low complexity deep learning algorithms for compensating atmospheric turbulence in the free space optical communication system," IET Optoelectron., vol. 16, no. 3, pp. 93–105, 2022.
[44]. M. A. Amirabadi, M. H. Kahaei, and S. A. Nezamalhosseini, "Deep learning based detection technique for FSO communication systems," IET Optoelectron., vol. 16, no. 3, pp. 93–105, 2022.
[45]. F. Aveta, S. Chan, N. Asfari, and H. Refai, "Atmospheric Turbulence Identification in a multi-user FSOC using Supervised Machine Learning," in Proc. 2021 IEEE 12th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON), New York, NY, USA, 2021, pp. 0964–0969.
[46]. Al-Kinani, C.-X. Wang, L. Zhou, and W. Zhang, "Optical wireless communication channel measurements and models," IEEE Commun. Surveys Tuts., vol. 20, no. 3, pp. 1939–1962, 3rd Quart. 2018.
[47]. F. Molisch, "Channel Sounding," in Wireless Communications, IEEE, 2011, pp. 145–164.
[48]. Recommendation ITU-R P.840-8: Attenuation due to clouds and fog, International Telecommunication Union, Radiocommunication Sector, Feb. 2019.
[49]. S. Alatawi, A. A. Youssef, M. Abaza, M. A. Uddin, and A. Mansour, "Effects of Atmospheric Turbulence on Optical Wireless Communication in NEOM Smart City," Photonics, vol. 9, no. 4, p. 262, 2022.
[50]. Y. Ho and S. Wookey, "The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling," IEEE Access, vol. 8, pp. 4806–4813, 2020.
[51]. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[52]. M. S. Santos, J. P. Soares, P. H. Abreu, H. Araujo, and J. Santos, "Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches [Research Frontier]," IEEE Comput. Intell. Mag., vol. 13, no. 4, pp. 59–76, Nov. 2018.
[53]. V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," Proc. IEEE, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
[54]. Y. Han *et al*., "Dynamic Neural Networks: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7436–7456, Nov. 2022, doi: 10.1109/TPAMI.2021.3117837.
[55]. J. Guo, C. L. P. Chen, Z. Liu, and X. Yang, "Dynamic Neural Network Structure: A Review for its Theories and Applications," IEEE Transactions on Neural Networks and Learning Systems, vol. 36, no. 3, pp. 4246–4266, Mar. 2025, doi: 10.1109/TNNLS.2024.3377194.
[56]. P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, 2004, doi: 10.1023/B:VISI.0000013087.49260.fb.
[57]. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," Neural Computation, vol. 3, no. 1, pp. 79–87, Mar. 1991, doi: 10.1162/neco.1991.3.1.79.

[58]. G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Weinberger, "Multi-scale dense networks for resource efficient image classification," in Proc. Int. Conf. Learn. Represent. (ICLR), Vancouver, BC, Canada, Apr. 2018. [Online]. Available: https://openreview.net/forum?id=Hk2aImxAb.

[59]. J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in Advances in Neural Information Processing Systems, vol. 30, pp. 2178–2188, 2017.

[60]. M. Ghiassi and H. Saidane, "A dynamic architecture for artificial neural networks," *Neurocomputing*, vol. 63, pp. 397–413, 2005, doi: 10.1016/j.neucom.2004.03.014.

[61]. W. Roth *et al*., "Resource-efficient neural networks for embedded systems," *Journal of Machine Learning Research*, vol. 25, no. 50, pp. 1–51, 2024.

[62]. P. K. Singya, P. Shaik, N. Kumar, V. Bhatia, and M.-S. Alouini, "A survey on higher-order QAM constellations: Technical challenges, recent advances, and future trends," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 617–655, 2021, doi: 10.1109/OJCOMS.2021.3067384.

[63]. T. Wang, G. Yang, P. Chen, Z. Xu, M. Jiang, and Q. Ye, "A survey of applications of deep learning in radio signal modulation recognition," *Applied Sciences*, vol. 12, no. 23, p. 12052, 2022, doi: 10.3390/app122312052.

[64]. G. Huang, "Dynamic neural networks: Advantages and challenges," *National Science Review*, vol. 11, no. 8, Aug. 2024, Art. no. nwae088, doi: 10.1093/nsr/nwae088.

[65]. A. Alqahtani, N. Aldhafferi, Atta-ur-Rahman, K. Sultan, and M. A. A. Khan, "Adaptive communication: A systematic review," *Journal of Communications*, vol. 13, no. 7, pp. 357–367, 2018.

[66]. N. Shazeer *et al*., "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint* arXiv:1701.06538, 2017.

[67]. B. Wójcik, A. Devoto, K. Pustelnik, P. Minervini, and S. Scardapane, "Adaptive computation modules: Granular conditional computation for efficient inference," in *Proc. AAAI Conf. Artif. Intell.*, vol. 39, no. 20, pp. 21510–21518, 2025, doi: 10.1609/aaai.v39i20.35453.

[68]. S. Scardapane, A. Baiocchi, A. Devoto, V. Marsocci, P. Minervini, and J. Pomponi, "Conditional computation in neural networks: Principles and research trends," *Intelligenza Artificiale*, vol. 18, no. 1, pp. 175–190, 2024, doi: 10.3233/IA-240035.

[69]. S. Li and X. Feng, "Dynamic weight strategy of physics-informed neural networks for the 2D Navier-Stokes equations," *Entropy*, vol. 24, no. 9, p. 1254, Sep. 2022, doi: 10.3390/e24091254.4o

[70]. R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti, "META-DES.H: A dynamic ensemble selection technique using meta-learning and a dynamic weighting approach," in *Proc. 2015 Int. Joint Conf. Neural Netw. (IJCNN)*, Killarney, Ireland, 2015, pp. 1–8, doi: 10.1109/IJCNN.2015.7280594

[71]. **Tektronix, Inc.**, "Wi-Fi Overview: 802.11 Physical Layer and Transmitter Measurements," *Primer*, Tektronix, [Online]. Available: https://www.tek.com/en/documents/primer/wi-fi-overview-80211-physical-layer-and-transmitter-measurements. [Accessed: Apr. 16, 2025].