FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING
DEGREE PROGRAMME IN WIRELESS COMMUNICATIONS ENGINEERING

MASTER'S THESIS

POTENTIAL DEEP LEARNING APPROACHES FOR THE PHYSICAL LAYER

Author	Rajapakshage Nuwanthika Sandeepani Rajapaksha
Supervisor	Prof. Nandana Rajatheva
Second Examiner	Dr. Janaka V. Wijayakulasooriya

July 2019

Rajapaksha R. (2019) Potential Deep Learning Approaches for the Physical Layer. University of Oulu, Faculty of Information Technology and Electrical Engineering, Degree Programme in Wireless Communications Engineering, 59 p.

ABSTRACT

Deep learning based end-to-end learning of a communications system tries to optimize both transmitter and receiver blocks in a single process in an end-to-end manner, eliminating the need for artificial block structure of the conventional communications systems. Recently proposed concept of autoencoder based end-to-end communications is investigated in this thesis to validate its potential as an alternative to conventional block structured communications systems. A single user scenario in the additive white Gaussian noise (AWGN) channel is considered in this thesis. Autoencoder based systems are implemented equivalent to conventional communications systems and bit error rate (BER) performances of both systems are compared in different system settings.

Simulations show that the autoencoder outperforms equivalent uncoded binary phase shift keying (BPSK) system with a 2 dB margin to BPSK for a BER of 10^{-5}, and has comparable performance to uncoded quadrature phase shift keying (QPSK) system. Autoencoder implementations equivalent to coded BPSK have shown comparable BER performance to hard decision convolutional coding (CC) with less than 1 dB gap over the 0-10 dB E_b/N_0 range. Autoencoder is observed to have close performance to the conventional systems for higher code rates. Newly proposed autoencoder model as an alternative to coded systems with higher order modulations has shown that autoencoder is capable of learning better transmission mechanisms compared to the conventional systems adhering to the system parameters and resource constraints provided. Autoencoder equivalent of half-rate 16-quadrature amplitude modulation (16-QAM) system achieves a better performance with respect to hard decision CC over the 0-10 dB E_b/N_0 range, and a comparable performance to soft decision CC with a better BER in 0-4 dB E_b/N_0.

Comparable BER performance, lower processing complexity and low latency processing due to inherent parallel processing architecture, flexible structure and higher learning capacity are identified as advantages of the autoencoder based systems which show their potential and feasibility as an alternative to conventional communications systems.

Keywords: autoencoder, deep learning, end-to-end learning, neural networks, communications, channel coding, modulation.

TABLE OF CONTENTS

ABSTRACT
TABLE OF CONTENTS
FOREWORD
LIST OF ABBREVIATIONS AND SYMBOLS
1 INTRODUCTION 7
1.1 Motivation and Thesis Objectives 8
1.2 Thesis Structure 9
2 BACKGROUND AND LITERATURE 10
2.1 Potential of Deep Learning for the Physical Layer 10
2.2 Deep Learning Basics 11
2.3 Deep Learning Libraries 12
2.4 Literature Review 13
2.4.1 Deep Learning based Block Structured Communications 13
2.4.2 Deep Learning based End-to-End Communications 15
3 END-TO-END LEARNING OF UNCODED SYSTEMS 18
3.1 End-to-End Learning of a Communications System 18
3.1.1 System Model 18
3.1.2 Autoencoder for End-to-End Learning 19
3.2 Autoencoder Implementation for Uncoded Communications Systems 20
3.3 Results and Analysis 21
4 END-TO-END LEARNING OF CODED SYSTEMS 26
4.1 System Model 26
4.1.1 Baseline for Comparison 27
4.2 Autoencoder Implementation for Coded Communications Systems with BPSK Modulation for AWGN Channel 27
4.2.1 Implementation 27
4.2.2 Results and Analysis 28
4.3 Autoencoder Implementation for Coded Communications Systems with Higher Order Modulations 35
4.3.1 Implementation 35
4.3.2 Effect of Model Layout and Hyperparameter Tuning for the Performance 37
4.3.3 Results and Analysis 38
4.3.4 Processing Complexity 43
4.3.5 Comparison with 5G Channel Coding and Modulation Schemes 45
5 CONCLUSION AND FUTURE WORK 46
5.1 Summary and Conclusion 46
5.2 Future Work 48
6 REFERENCES 49
7 APPENDICES 52

FOREWORD

This thesis is focused on potential deep learning approaches for the physical layer as a part of the High5 and MOSSAF projects at the Center for Wireless Communications (CWC) of University of Oulu, Finland. I would like to express my sincere gratitude to my supervisor and mentor Prof. Nandana Rajatheva for the guidance, support, inspiration and encouragement given throughout the period of my master studies. I am also grateful to Academy Prof. Matti Latva-aho for providing me the opportunity to join and contribute to the High5 and MOSSAF projects. Also I would like to thank project manager Dr. Pekka Pirinen and the other colleagues at CWC for their support in my research. I would also like to express my gratitude to Matti Isohookana, the coordinator of Double Degree Master's Program, for his support and guidance throughout the past year.

I am also thankful to Dr. Janaka V. Wijayakulasooriya, my supervisor from University of Peradeniya, Sri Lanka, for the support given. Also I would like to thank all the lecturers from University of Peradeniya for their contribution in making the inaugural Double Degree Master's Programme a success.

Finally I would like to express my sincere gratitude to my mother, father and brother for their immense love, support and encouragement provided throughout my life.

Oulu, 30th July, 2019

Rajapakshage Nuwanthika Sandeepani Rajapaksha

LIST OF ABBREVIATIONS AND SYMBOLS

Acronyms

1D	One-Dimensional
2D	Two-Dimensional
3GPP	3rd Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation
ANN	Artificial Neural Network
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BAIR	Berkeley Artificial Intelligence Research
BER	Bit Error Rate
BLER	Block Error Rate
BPSK	Binary Phase-Shift Keying
CC	Convolutional Coding
CNN	Convolutional Neural Networks
CP	Cyclic Prefix
CPU	Central Processing Unit
CSI	Channel State Information
DL	Deep Learning
DNN	Deep Neural Network
GAN	Generative Adversarial Network
GD	Gradient Descent
GPU	Graphics Processing Units
LDPC	Low Density Parity Check
LTE	Long-Term Evolution
MASK	M-ary Amplitude Shift-Keying
MER	Message Error Rate
MFSK	M-ary Frequency Shift-Keying
ML	Machine Learning
MLP	Multi Layer Perceptrons
MMSE	Minimum Mean Square Error
MSE	Mean Squared Error
NLP	Natural Language Processing
NN	Neural Network
NR	New Radio
OFDM	Orthogonal Frequency-Division Multiplexing
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
ReLU	Rectified Linear Unit
RTN	Radio Transformer Network
SDR	Software-Defined Radio
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise Ratio
SVM	Support Vector Machine

TBCC	Tail Biting Convolutional Codes
TPU	Tensor Processing Unit
URLLC	Ultra-Reliable Low-Latency Communication

Symbols

$\mathbb{C}^n$	Set of complex n-vectors
$\mathbb{M}$	Set of M messages
E_b/N_0	Energy per bit to noise spectral density ratio
K	Information block size
M	Number of messages
M_{mod}	Modulation order
N	Size of the channel coded block
P_e	Message error rate (MER)
R	Rate (communication rate or code rate as applicable)
k	Number of bits in in message s
k_{mod}	Number of bits per codeword when modulated with M_{mod} order modulation
n	Number of complex channel uses for transmitting message s
s	Transmit message
$\hat{s}$	Estimation of message s
$\mathbf{x}$	Transmitted signal vector
$\mathbf{y}$	Received signal vector
β	Noise variance of the AWGN channel

1 INTRODUCTION

Wireless networks and related services have become critical and fundamental building blocks in the modern digitized world which have changed the way we live, work and communicate with each other. Emergence of many unprecedented services and applications such as autonomous vehicles, remote medical diagnostics and surgeries, smart cities and factories, artificial intelligence based personal assistants etc. are challenging the traditional communication mechanisms and approaches in terms of latency, reliability, energy efficiency, flexibility and connection density. Catering the stringent requirements arising from those different verticals requires a greater need for wireless system research with novel architectures, approaches and algorithms in almost all the layers of a communications system. Newly initiated fifth generation (5G) of mobile communication technology is expected to cater for these requirements revolutionizing everything so far in wireless-enabled applications [1].

As said, 5G brings most stringent requirements when catering to the advanced applications and services which will be supported by it. For example, ultra-reliable low-latency communication (URLLC), one category out of the three service categories defined in 5G, which perhaps may be the most challenging as it needs to meet two challenging and contradicting requirements: low latency and ultra-high reliability, requires end-to-end latency in range of 10 ms and very high reliability with 10^{-5} bit error rate (BER) in 1 ms period [2]. High reliability means that the channel estimation accuracy should be improved since the channel coding gain is small for the short packet lengths so that the loss, if any, caused by the channel estimation should be prevented as much as possible. It is to be achieved by advanced channel estimation techniques and by adding more resources to the pilots which again raises concerns with latency requirements as more pilots result in control overhead which affect the throughput and hence latency of the communications. Another concern is faster signal processing at the transmitter and receiver to achieve low latency requirements of URLLC. Therefore, for successful implementation of URLLC systems, all these factors need to be taken into consideration which requires novel architectures, approaches and algorithms in almost all the layers of the communication system.

Communications field is very rich of expert knowledge based on statistics, information theory and solid mathematical modelling capable of modelling channels [3], optimal signalling and detection schemes for reliable data transfer compensating for various hardware imperfections etc. especially for the physical layer [4]. However, existing conventional communication theories exhibit several inherent limitations in fulfilling the large data and ultra high rate communication requirements in complex situations such as channel modelling in complex scenarios, fast and effective signal processing in low latency systems such as URLLC, limited and sub-optimal block structures due to the fixed block structure of the communication systems etc. In recent history, there has been an increasing interest in deep learning approaches for the physical layer implementations due to certain advantages they possess which could be useful in overcoming the above challenges.

1.1 Motivation and Thesis Objectives

Fundamental requirement of a communications system is to reliably transmit a message from a source to a destination over a channel by the use of a transmitter and a receiver. In order to achieve an optimal solution in practice, transmitter and receiver are typically divided into chain of multiple independent blocks, each responsible for a specific sub-task such as source/channel coding, modulation/demodulation, channel estimation and equalization etc. [4]. Although this block structure enables individual analysis, optimization and controlling of each block, it is not clear that individually optimized processing blocks achieve the best possible end-to-end performance and that approach is known to be sub-optimal in certain instances [5].

On the other hand, a deep learning based communications system follows the initial definition of a communications system and tries to jointly optimize transmitter and receiver in an end-to-end manner without having a defined block structure [6], [5]. Such a simple and straightforward structure seems appealing to be implemented in practical systems with less computing complexity and processing delays, and with less power consumption, given that it can provide equal or better performance than the existing systems.

Based on the above motivations, the objective of this thesis is to study and develop deep neural network (DNN) based models for a communications system in an end-to-end manner with the intention of achieving similar or better performance in terms of block error rate (BLER)/ BER or reliability compared to the conventional communication algorithms, at the same time reducing the processing complexity and latency. The target is to see the potential of deep learning based approaches to replace or assist the conventional communication algorithm implementations in the physical layer when trying to achieving the 5G specifications.

Simple and flexible structure of autoencoder based communications system [5] and its capability to learn adjusting to the given channel model and system parameters has inspired us to study about the autoencoder based end-to-learning of communications systems. In this thesis, we extend the previous research done in autoencoder based end-to-end learning of communications systems comparing the performance of deep learning models with their equivalent conventional implementations with different modulation schemes and code rates to understand the potential of deep learning based end-to-end communications as an alternative to conventional implementations. Scope of research conducted for this thesis is limited to single user implementations over the AWGN channel.

Main objectives of the thesis can be stated as follows.

- Perform a literature review about the 5G physical layer requirements and specifications, get an understanding about the challenges faced by current technologies and implementations.
- Conduct a thorough literature review on deep learning concepts for the physical layer and identify appropriate deep learning approaches as alternatives for the conventional communications systems.

- Study about the autoencoder based end-to-end learning of communications systems and implement the basic system proposed by [5] for a simple transmitter-receiver system.
- Study the performance of the autoencoder based system as an alternative to uncoded communications systems comparing performance for different modulation schemes.
- Study the performance of the autoencoder based systems in comparison to equivalent conventional coded communications systems.
- Improve the existing autoencoder model or develop a new model to enable implementing systems equivalent to coded systems with different higher order modulation with comparable performance to the existing conventional systems.
- Analyse the processing complexity of autoencoder based systems and equivalent conventional systems in order to get an overall understanding for comparing the performance capability and computational complexity which are important parameters when proposing deep learning based implementations as alternatives to conventional communications systems.

1.2 Thesis Structure

The thesis is structured into five chapters. In this chapter we have given an overview about requirements and challenges which are needed to be addressed by future communications systems and have discussed our motivations to look into deep learning based approaches for the physical layer communication blocks. In the second chapter we present the background and literature related to the thesis work. There we discuss the potential of deep learning for the physical layer in detail, along with deep learning basics which are relevant in the context of studied literature and the thesis work. Then we present the literature review, explaining the existing work on deep learning based block structured communications and deep learning based end-to-end communications.

In the third chapter, we discuss the autoencoder concept for end-to-end learning of communications systems and analyse the performance of the autoencoder based end-to-end system proposed by [5] in comparison to conventional uncoded systems with different modulation schemes. In the fourth chapter we analyse the performance of the autoencoder based end-to-end learning system in comparison to conventional coded systems with different modulation schemes. There, we also present a new autoencoder model to cater for implementing equivalent autoencoder counterpart of coded systems with higher order modulation schemes with a comparable BER performance to the baseline systems compared. In the last chapter we present the conclusions of our research findings along with future directions for improvements.

2 BACKGROUND AND LITERATURE

There have been attempts to apply machine learning (ML) for the physical layer for few decades where researchers have proposed ML based algorithms for different sub tasks in the physical layer such as modulation recognition [7], [8], encoding and decoding [9], [10], channel modelling and identification [11], channel estimation and equalization [12], [13], [14] etc. However, it is seen that ML has not been commercially used due to the fact that ML algorithms do not have enough learning capability to cater for the complex task of handling physical channels.

It is believed that introduction of deep learning (DL) to the physical layer could bring further performance improvements to the existing ML approaches, eliminating the limitations faced by conventional ML algorithms, due to the characteristics it has such as deep modularization which enhances feature extraction and structural flexibility to a great extent compared to ML algorithms [15]. Specifically, DL-based systems could be used to automatically learn features from raw data instead of manual feature extraction where flexible adjustment of model structures via hyperparameter tuning is possible in order to optimize end-to-end performance of the system.

In this chapter we discuss the potential of applying DL for the physical layer which has created a great interest among the research community to study DL-based approaches for the physical layer. We then present a basic overview about the basic DL concepts where a detailed description of different DL concepts is available in Appendix 1. An overview of different DL libraries is also presented. Finally, we present a detailed overview of selected literature which have proposed new DL-based approaches in the physical layer of the communication systems.

2.1 Potential of Deep Learning for the Physical Layer

- Signal processing algorithms in communications are mostly based on statistics and information theory and are often proved to be optimal for tractable mathematical models. However, practical systems have many imperfections and non-linearities which may be difficult to fully capture from mathematical models [5].

  On the other hand, deep networks have been proven to be universal function approximators [16] with a better learning ability and thus can be used to model communications systems despite complex channel conditions and hardware imperfections that are mathematically difficult to model. The "learned" algorithms in DL-based communication systems are represented by the weights learned in DL models which have been optimized for end-to-end performance through training, instead of requiring well defined mathematical models or expert algorithms [15].

- DL approaches essentially require to cater for handling large amounts of data due to their nature of distributed and parallel processing architectures, which enables computational speed and processing capacity. DL systems have a great potential in producing a considerable computational throughput through parallelized processing architectures [15].

- Since the execution of neural networks (NNs) can be highly parallelized on concurrent architectures and implemented with low-precision data types, it is

observed that these "learned" algorithms could be executed faster and at lower energy cost than their manually "programmed" counterparts [5]. Parallel processing architectures with distributed memory architectures such as graphics processing units (GPUs) and specialized chips for NN inferences, are proved to be very energy efficient and capable of providing considerable computational throughput when fully utilized by parallel implementations [5].

- DL based communication systems do not require the conventional block structure to achieve the global performance as they target optimizing the end-to-end performance instead of explicitly optimizing the individual blocks. Thus, the artificial block based structure will be not required if the whole communications system is based on a DL approach, and there is potential to improve the overall performance if such an end-to-end system is implemented.

- Recent development of dedicated DL libraries, tools and frameworks such as TensorFlow [17], PyTorch [18], Caffe [19] etc. enable fast experimentation and prototyping of DL models and quick and easy deployment of them enabling applying DL-based approaches for a wide variety of application areas.

2.2 Deep Learning Basics

DL is a branch of ML which enables an algorithm to predict, classify or make decisions based on data without being explicitly programmed. DL uses multiple layered structures and nonlinear processing units to hierarchically extract higher level features from raw input data in order to optimize a given target objective in contrast to the conventional ML algorithms which depend on manually extracted features by domain experts, thus has a wide learning capacity. DL expands in supervised, unsupervised and reinforcement learning domains similar to the ML algorithms. While DNNs are the most known DL models, other deep architectures such as deep Gaussian processes, neural processes, and deep random forests can also be regarded as deep models which consist of multiple layered structures [20].

The history of DL goes back to 1940s-1960s where the development of theories of biological learning and implementations of the first models such as the perceptron which allowed training of a single neuron were happened [21]. However, it was only in late 1980s where NNs gained interest after Rumelhart et al. [22] showed the possibility of training NNs consisting of one or two hidden layers using back propagation [20]. DL has gained a much wider interest and a rapid growth in the past 15-20 years due to the technological advancements like GPUs, software libraries and due to big data etc. which have eliminated computational limitations of training DL models, resulting in using DL-based approaches in a wide variety of applications [20].

Feedforward neural networks, convolutional neural networks (CNNs), autoencoders and generative adversarial networks (GANs) are some of the different types of DL models which are mainly used in literature under the scope of this thesis. A detailed overview about those DL concepts are presented in Appendix 1 along with an overview about the deep network training process. There we have mainly referred [21] for the theoretical parts related to DL concepts.

2.3 Deep Learning Libraries

Building a DL model from the scratch is a complex task and requires a great effort as it requires definitions of forwarding behaviours and gradient propagation operations at each layer and implementing efficient and fast optimization algorithms for model training, in addition to CUDA coding for GPU parellelization. In recent years, DL has gained a great momentum and popularity being used in quite many application areas such as image and video recognition, speech recognition and natural language processing (NLP) etc. This continuously-growing usage and popularity of DL has resulted in development of numerous tools, algorithms and dedicated libraries which make it easy to build and train large NNs. Most of these tools allow high level algorithm definition in various programming languages or configuration files, automatic differentiation of training loss functions through arbitrarily large networks, and compilation of the network's forwards and backwards passes into hardware optimized concurrent dense matrix algebra kernels [5]. They are built with massively parallel GPU architectures enabling GPU acceleration which makes faster processing of model training routines of large networks with huge amounts of data. A brief summary of some of the widely used libraries is given below.

- TensorFlow

  Created by the Google Brain team, TensorFlow is an open source library for numerical computation and large-scale machine learning that operates at large scale and in heterogenous environments [17]. TensorFlow uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multicore cetnral processing units (CPUs), general-purpose GPUs, and custom-designed application-specific integrated circuits (ASICs) known as Tensor processing units (TPUs) [17]. TensorFlow supports multiple languages to create DL models. Some of the languages that it supports are Python, C++, Java, Go, R. Currently, the best-supported client language is Python with detailed documentation and tutorials. Keras [23], Luminoth and TensorLayer are some of the dedicated DL toolboxes which are built upon TensorFlow, which provide higher-level programming interfaces. Keras is the main tool which we have used to implement the DL models we have proposed in this thesis, as it has a very user friendly and highly customizable interface which enables quick and easy prototyping for experimenting.

- PyTorch

  PyTorch [18] is a Python based open source DL library inspired by Torch. It is a framework built to be flexible and modular for research, with the stability and support needed for production deployment. PyTorch has been primarily developed by Facebook's artificial intelligence research group. PyTorch is one of the preferred DL research platforms built to provide maximum flexibility and speed. It is known for providing two of the most high-level features; namely, tensor computations with strong GPU acceleration support and building deep neural networks on a tape-based autograd systems designed for immediate and python-like execution. PyTorch has a growing popularity among the research community since building NNs in PyTorch is straightforward.

- Caffe

  Caffe is a dedicated DL framework made with expression, speed, and modularity in mind. It is developed by Berkeley artificial intelligence research (BAIR) [19] and by community contributors. It allows to train NNs on multiple GPUs within distributed systems, and supports DL implementations on mobile operation systems, such as iOS and Android [20].

2.4 Literature Review

Recent studies have proposed DL based approaches to improve the performance of the communications systems, motivated by the potential of DL for the physical layer applications (Section 2.1), which are based on and enabled by the concepts and technologies as discussed in Appendix 1. In literature, two main approaches are proposed, where DL is either used to enhance and optimize certain blocks of the conventional communication system as an alternative to the existing processing algorithms in each block (such as modulation recognition, channel encoding and decoding, channel estimation and detection etc.) or to completely replace the total block based communication system with a DL based architecture which optimizes the end-to-end performance of the system without a specific block structure like in conventional systems. In this section, we present an overview of selected literature which has DL based applications that cover the aforementioned two approaches.

2.4.1 Deep Learning based Block Structured Communications

As noted earlier, the fundamental requirement of a communications system is reliable transmission of a message from source to destination over a communications channel. In practice, a conventional communications system is constructed as a block structure in order to achieve an optimal solution for this task where different algorithms which are based on expert knowledge have been developed over decades in order to optimize each individual block of the communications system. The structure of a standard communications system is shown in Figure 2.1 which consists of source/channel coding and decoding, modulation and demodulation, channel estimation and signal detection.

Over the years, studies have proposed to use conventional machine learning approaches such as support vector machines (SVMs) and small feedforward NNs as alternative implementations to cater for those individual tasks [15]. With the development of DL algorithms and architectures, in recent years DL based approaches have been introduced into different processing blocks of the communications systems as alternatives to conventional algorithms such as modulation recognition [5], channel encoding and decoding [24], [25], [26], [27], [28], [29] and channel estimation and detection [30], [31], [32], [33].

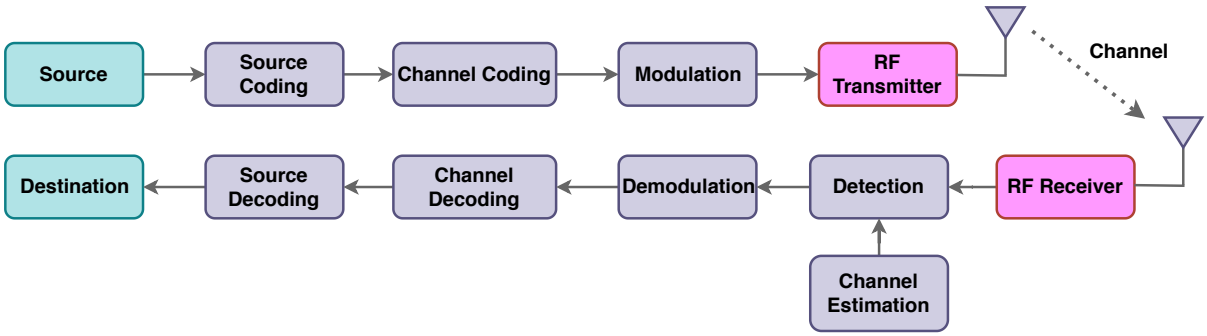

Figure 2.1. Structure of a classical communications system.

These DL based algorithms are expected to be more effective in complex communications scenarios where the learning capability of DL can be leveraged to adapt to different complex channel conditions, complex operating environments etc. Also DL based approaches would be more useful in instances where reduced the computational complexity and processing overhead are preferred. In this section, we present some of the examples which use DL applications for different processing blocks such as signal detection and modulation recognition.

Deep Learning based Joint Channel Estimation and Signal Detection

In a conventional communications system, channel estimation and signal detection are two separate procedures executed at the receiver. Channel state information (CSI) is first estimated explicitly with the aid of the pilots prior to the detection of transmit symbols, and the estimated CSI is then used to detect/recover the transmitted symbols at the receiver. In [32], a DL based approach for joint channel estimation and signal detection in orthogonal frequency-division multiplexing (OFDM) systems is presented which uses DL to handle wireless OFDM channels in an end-to-end manner. In contrast to the existing conventional approach of explicit CSI estimation and symbol detection, the proposed DL based receiver implicitly estimates the CSI and recovers the transmitted symbols directly which is implemented as a five-layer fully connected DNN. They have initially trained an offline DNN model with generated data based on channel statistics and the trained model is then used to recover the online transmitted data directly. Their results have shown that the DL based approach performs in comparable to the minimum mean square error (MMSE) estimator. Proposed method shown to be more robust when a lesser number of pilots are used, cyclic prefix (CP) is omitted and when nonlinear clipping noise exists.

Deep Learning based Modulation Recognition

Modulation recognition which has the goal of differentiating different modulation schemes of received signals is an important task in the field of communications as it facilitates communication among the different communications systems, or interfering and monitoring enemies for military use. Modulation classification task has been studied over decades through the approach of expert feature engineering and either analytic decision trees or trained discrimination methods operating on a compact feature space, such as support vector machines, random forests, or small feedforward NNs. These approaches

consist of several common procedures such as preprocessing, feature extraction, and classification. Some researches have presented some advanced approaches using pattern recognition on expert feature maps, such as the spectral coherence function or α-profile, combined with NN-based classification.

In [34], an NN architecture is proposed as a modulation classifier to distinguish noise-corrupted band-limited modulated signals from 13 types of digital and analog modulation schemes. This approach manually extracts features from the signals including features which characterize digital or analog modulations and the basic parameters of signals such as amplitude, phase and frequency. Those features are fed to a four-layer NN to discriminate the different modulation schemes and two more two-layer NNs are used to estimate the levels of the M-ary amplitude shift-keying (MASK) and the M-ary frequency shift-keying (MFSK) modulations. The performance of these kind of feature engineered approaches strongly depends on the extracted features due to the limited learning ability of conventional NNs [15]. In several recent studies, the well admired learning capacity of DL is exploited to overcome the aforementioned limitations of feature based modulation classification approaches.

In [5], authors highlight the possibility to replace the artificial feature extraction with automatic learning of features from the raw data to optimize end-to-end performance, by using the superior learning capability of DNNs. They have proposed a convolutional neural network (CNN) based approach for modulation classification of single carrier modulation schemes based on sampled radio frequency time-series data. The CNN classifier is trained using a dataset of 1.2M sequences of 128 complex-valued baseband in-phase and quadrature (I,Q) samples corresponding to 10 different digital and analog single-carrier modulation schemes which have gone through a wireless channel with multipath fading effects and clock and carrier rate offset. Their results have shown that the CNN-based modulation classifier outperforms two other approaches: extreme gradient boosting with 1000 estimators and a single scikit-learn tree working on the extracted expert features, mainly in the low to medium signal-to-noise ratio (SNR) range where as in the high SNR range, CNN and boosted tree performance are similar.

2.4.2 Deep Learning based End-to-End Communications

In the previous section, we discussed several DL based approaches which are used as alternatives for one or two processing blocks of the conventional block structured communications system. However, when looking back to the original requirement of a communications system of transmitting a message from source to destination over a channel, even though the block structure enables individual analysis and controlling of each block, it can not be guaranteed that optimization of each block will always result in global optimization for the communication problem because end-to-end performance improvements can be achieved by joint optimization of two or more blocks.

A novel DL based concept has been introduced in recent history based on this thought process, which reformulates the communication task as an end-to-end reconstruction optimization task where the artificial block structure of the conventional communications system is no longer required. This novel concept is based on implementing the end-to-end communications system by an autoencoder system and the initial studies have shown that it has comparable performance to the conventional systems and also has shown that

the end-to-end method has great potential to be a universal solution for different channel models. In this section we discuss about that newly introduced concept of autoencoder based end-to-end communications and present details about some of very recent studies based on that.

Autoencoder based End-to-End Communications

Using the autoencoder concept for the communications system was first introduced in [6] and [5]. In [5], communications system is interpreted as an autoencoder and a new way of communications system design as an end-to-end reconstruction task which jointly optimizes the transmitter and receiver components in a single process is presented. They have shown that it is possible to learn transmitter and receiver implementations for a given channel model which are optimized for a desired loss function such as minimizing the BLER, using a DL model. They have implemented the transmitter, channel and the receiver as one DNN that can be trained as an autoencoder. The transmitter and receiver are represented as fully connected feedforward NNs and the AWGN channel between them is represented as a noise layer with the desired noise variance. Thus, the communications system can be seen as an autoencoder which tries to learn from the message s out of M possible messages $s \in \mathbb{M} = \{1, 2, ..., M\}$, to generate the representation of the transmitted signal $\mathbf{x}$ which is robust against the communication channel. At the receiver, the original message s is reconstructed as $\hat{s}$ with the lowest possible error by learning from the received signal $\mathbf{y}$.

The whole network is trained end-to-end in order to achieve BLER performance. Their results have shown that the autoencoder based communications system achieves a comparable or better performance than standard BPSK with Hamming code, thus indicating that the system has learnt a joint coding and modulation scheme to minimize the BLER over the AWGN channel. Potential of application of the autoencoder based approach for channel models and loss functions where optimal solutions are not known is also noted in their study.

The potential of introducing communication expert knowledge to the autoencoder model which improves its performance, enables adjusting the DL architecture to accommodate different communication scenarios, and accelerates the DL model training phase is also shown in [5] by introducing the concept of radio transformer networks (RTNs). As noted by [5], RTNs enable carrying out predefined correction algorithms or "transformers" at the receiver (such as multiplication by a complex-valued number, convolution with a vector etc.) which can be used to reverse the channel effect occurred during signal transmission over the imperfect channel. The transformation layer can be fed with parameters learned by another NN and they can be integrated into the receiver in order perform symbol detection in a more effective manner by enabling integration of communication knowledge into the DL system. The authors have shown performance improvements and fast convergence of the model training than the plain autoencoder when RTN is used.

In [35], a more recent study carried out inspired by the initial findings of [5], the concept of autoencoder based end-to-end communications system is extended to an actual implementation showing the feasibility of over-the-air transmission. They have modeled, trained and run a complete communications system consisting of NNs using unsynchronized off-the-shelf software-defined radios (SDRs) and open source DL software

libraries. The limitation of short block lengths faced by the autoencoder models has been overcome by implementing mechanisms for continuous data transmission and receiver synchronization where a frame synchronization module based on another NN is implemented at the receiver to cater for the receiver synchronization. A two step training procedure based on transfer learning is used to overcome training the model over actual channels by finetuning the receiver part of the autoencoder to capture the effects of the actual channel including the hardware imperfections which are not initially included in the model. Comparison of the BLER performance of the "learned" system with that of a practical baseline have shown a comparable performance close to 1 dB. The study has thus validated the potential of actual implementation of autoencoder based communication systems.

Autoencoder for Multi-User

Extension of the autoencoder model to an adversarial network of multiple transmitter and receiver pairs competing for capacity is also presented in [5]. Simple two-user scenario is considered where the transmitter-receiver pair of each user attempt to communicate simultaneously over the same channel which leads to the interference channel where finding the best signalling scheme is known to be a well known long-standing research problem. In the two-user scenario, the overall system is trained to achieve conflicting goals at the receiver side, where each user tries to optimize the system to transmit their own messages in best possible accuracy. The system is represented as a multiple input-output NN and both transmitter-receiver pairs are jointly optimized with respect to a common performance metric, minimizing the weighted sum of both losses denoted by $\mathbf{L} = \alpha\mathbf{L_1} + (\mathbf{1} - \alpha)\mathbf{L_2}$ for some $\alpha \in [0, 1]$. The results show that the autoencoder system obtains similar of better BLER performance at the same communication rate than the uncoded quadrature amplitude modulation (QAM) schemes, thus validates the potential of application of autoencoder model in multi-user cases as well.

Over-the-Air Communications based on Adversarial Networks

In [36], channel autoencoder model is extended to enable end-to-end learning of the communications system when the channel response is unknown or cannot be easily modelled in a closed-form analytical expression. Previously, over-the-air channel autoencoders were implemented in two phases: initial pre-training based on a closed-form channel model to match the expected deployment scenario and fine-tuning the receiver part of the model to capture the actual channel effects [35]. In finetuning stage, optimization of only the receiver side of the network is possible as it is unable to back propagate through the black-box void of the radio channel. This limitation has been overcome in [36] by introducing an adversarial approach for channel response approximation where a learning based approach which does not require an analytic model for channel impairments is implemented. It is based on generative adversarial networks where the model tries to jointly optimize the two tasks of: 1) approximating the channel response of any arbitrary communications system, 2) learning an optimal encoding and decoding scheme that optimizes a given performance metric such as BLER. Their results show that such a model can result in an effective communications system which can achieve robust performance without the need of a close-form channel model or implementation.

3 END-TO-END LEARNING OF UNCODED SYSTEMS

As discussed in Chapter 2, end-to-end learning of a communications system using DL based autoencoder concept has drawn interest in recent research due to its simplicity, flexibility and its potential of adapting to complex channel models and practical system imperfections. In this thesis, we have extended the existing research on autoencoder based end-to-end learning of communications system in order to investigate its performance in different system configurations in order to understand the potential of autoencoder based end-to-end learning of communications systems.

In this chapter, we investigate the performance of autoencoder based end-to-end communications system in comparison with a conventional communications system when there is no channel coding applied. Thus, in this chapter we compare the performance of the autoencoder based system with conventional uncoded BPSK and QPSK performance over the AWGN channel.

First we explain the autoencoder concept and its application for end-to-end learning of communication systems as proposed by [5] formulating the system model and the autoencoder model layout. Then the implementation details are presented in Section 3.2 followed by results obtained for various configurations which is presented in 3.3. We have used the BER metric to compare the performance of the autoencoder and conventional implementations.

3.1 End-to-End Learning of a Communications System

3.1.1 System Model

A basic communications system consists of a transmitter, a channel and a receiver as shown in Figure 3.1. The transmitter wants to communicate a message s out of M possible messages $s \in \mathbb{M} = \{1, 2, ..., M\}$ to the receiver using n uses of the channel. It transmits a vector $\mathbf{x}$ of n complex symbols over the channel to send the message s to the receiver. Physical implementation at the transmitter imposes power constraints on $\mathbf{x}$ such as an energy constraint $\mathbf{x}_2^2 \leq n$ or an average power constraint $\mathbb{E}[x_i^2] \leq 1 \forall i$.

Figure 3.1. Structure of a simple communications system.

Each message s can be represented in $k = log_2(M)$ number of bits. Thus, the system operates in $R = k/n$ [bits/channel use] communication rate. The channel causes distortions to the transmitted symbols and at the receiver upon reception of signal $\mathbf{y} \in \mathbb{C}^n$, the receiver produces the estimate $\hat{s}$ of originally transmitted message s. The message error rate (MER) P_e can be defined as

$$P_e = \frac{1}{M} \sum_s Pr(\hat{s} \neq s|s). \tag{1}$$

3.1.2 Autoencoder for End-to-End Learning

The above-mentioned simple communications system was first proposed to be interpreted as an autoencoder in [5]. As described in Appendix 1.C, an autoencoder is a type of ANN that tries to reconstruct its input at the output in an unsupervised manner. Thus, the communications system can be thought as an autoencoder which tries to reconstruct the transmit message at the receiver with a best possible minimum error. The encoder function and decoder function of the autoencoder model can be thought as the transmitter and receiver blocks of the system respectively. An autoencoder architecture which can be used to end-to-end learning of a communications system is shown in Figure 3.2.

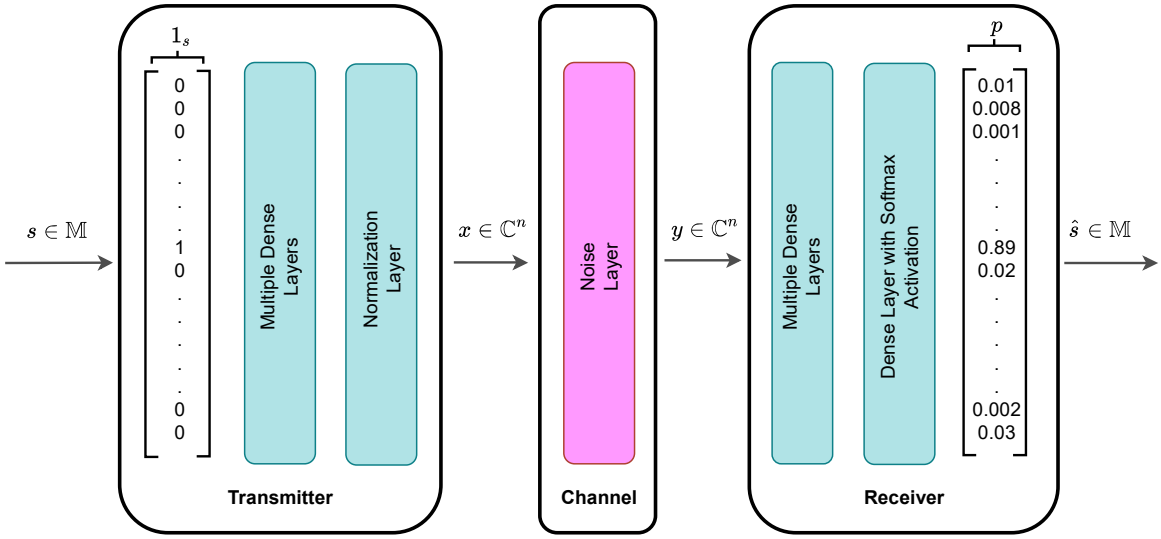

Figure 3.2. A communications system represented as an autoencoder.

There, the transmitter is implemented as a feedforward NN with multiple dense layers followed by a normalization layer to meet the physical constraints of transmit vector $\mathbf{x}$. Input s to the NN is an M-dimensional *one-hot encoded* vector $\mathbf{1}_s \in \mathbb{R}^M$ i.e., an M-dimensional vector, the s^{th} element of which is equal to one and zero otherwise. Transmitter output is a $2n$-dimensional vector which corresponds to n complex symbols transmitted in n channel uses by considering one half as real part and the other as the imaginary part. The channel is represented by an additive noise layer with a fixed variance $\beta = (2RE_b/N_0)^{-1}$, where E_b/N_0 denotes the energy per bit (E_b) to noise power spectral density (N_0) ratio. The receiver is also implemented as a feedforward NN with a single or multiple dense layers followed by an output layer with softmax activation whose output $\mathbf{p} \in (0,1)^M$ is a probability vector over all possible messages. The decoded message $\hat{s}$ corresponds to the element index of $\mathbf{p}$ which has the highest probability.

This autoencoder can be trained end-to-end using stochastic gradient descent (SGD) or any other suitable optimization method on the set of all possible messages $s \in \mathbb{M}$ using the categorical cross-entropy loss function.

3.2 Autoencoder Implementation for Uncoded Communications Systems

When investigating the potential of autoencoder implementation as an end-to-end communications system, we have first compared its performance with a simple transmitter-receiver structure without any channel coding applied. This section presents the implementation and the results obtained for the autoencoder based transmitter receiver system in comparison with the standard communications system with different modulation schemes.

We have implemented the autoencoder architecture proposed by [5] with slight modifications. The autoencoder models have been trained for different message alphabet sizes (M) with different communication rates and are compared with baseline BPSK and QPSK systems accordingly. The autoencoder was implemented as a fully connected feedforward NN. Different autoencoder architectures with different number of hidden layers and different activation functions were investigated. Table 3.1 lists out the activation function types and output dimensions in each layer of the selected optimum model layout where the autoencoder is constructed by sequentially combining the layers in the order listed in the table.

Table 3.1. Layout of the autoencoder model

Layer	Output dimensions
Input	M
Dense-ReLU	M
Dense-ReLU	M
Dense-Linear	2n
Normalization	2n
Noise	2n
Dense-ReLU	M
Dense-ReLU	M
Dense-Softmax	M

Layers 1-5 compose the transmitter side of the system where the energy constraint of the transmit signals is guaranteed by the normalization layer at the end. Layers 7-9 compose the receiver side of the system where estimated message can be found from the output of the softmax layer. Noise layer in-between the transmitter and receiver side of the system acts as the AWGN channel.

Autoencoder is trained end-to-end over the stochastic channel model using SGD method with the Adam optimizer with learning rate = 0.001. Following approaches were taken to select E_b/N_0 values for the AWGN channel during training:

- Training at a fixed E_b/N_0 value (i.e., 5 dB or 8 dB etc.)
- Picking E_b/N_0 values randomly from a predefined E_b/N_0 range for each training epoch
- Starting from a high E_b/N_0 value and gradually decreasing it along training epochs (i.e., starting from 8 dB and reduce by 2 dB after each 10 epochs)

Autoencoder model training and testing was implemented in Keras [23] with TensorFlow [23] as its backend. We have trained the models over 50 epochs using 1,000,000 randomly generated messages with E_b/N_0 values for AWGN channel in model training in three settings mentioned earlier. Testing the trained models were done with 1,000,000 different random messages for 0 dB to 8 dB E_b/N_0 range and their BER performance have been compared with the corresponding baseline systems.

We have tried out several autoencoder configurations which result in BPSK equivalent systems where communication rate $R = 1$ bit/channel use and QPSK equivalent systems where $R = 2$ bits/channel use. The message alphabet size M and number of channel uses n have been set accordingly in order to achieve the desired communication rate. Table 3.2 shows the autoencoder configuration parameters and their baseline systems which we have compared performance with. Total energy per message is kept same in both autoencoder system and baseline system in each scenario.

Table 3.2. Different autoencoder configuration parameters and their baseline systems used for performance comparison

Number of messages (M)	Complex channel uses per message (n)	Communication rate (R)	Equivalent modulation scheme
2	1	1 bit/channel use	BPSK
4	2		
8	3		
16	4		
4	1	2 bits/channel use	QPSK
16	2		
64	3		
256	4		

3.3 Results and Analysis

BER performance

Figure 3.3 shows the BER performance of $R = 1$ systems compared with theoretical AWGN BER performance of their baseline BPSK scheme. It seems that all four autoencoder configurations have equal or better BER performance across almost full E_b/N_0 range except from 0 dB to 2 dB where some autoencoder systems have slightly higher BER than BPSK system. It is interesting to see that the BER performance improves when the message alphabet size and number of channel uses per message increases even though communication rate is the same in all models. That is probably because the transmitted messages get some sort of temporal encoding since multiple channel uses are used to transmit a single message. When the number of channel uses per message increases, more flexibility and degree of freedom is there to formulate the

transmit symbols adjusting to the channel distortions and hence transmit symbols are more tolerant to errors and can be recovered at the receiver with less errors.

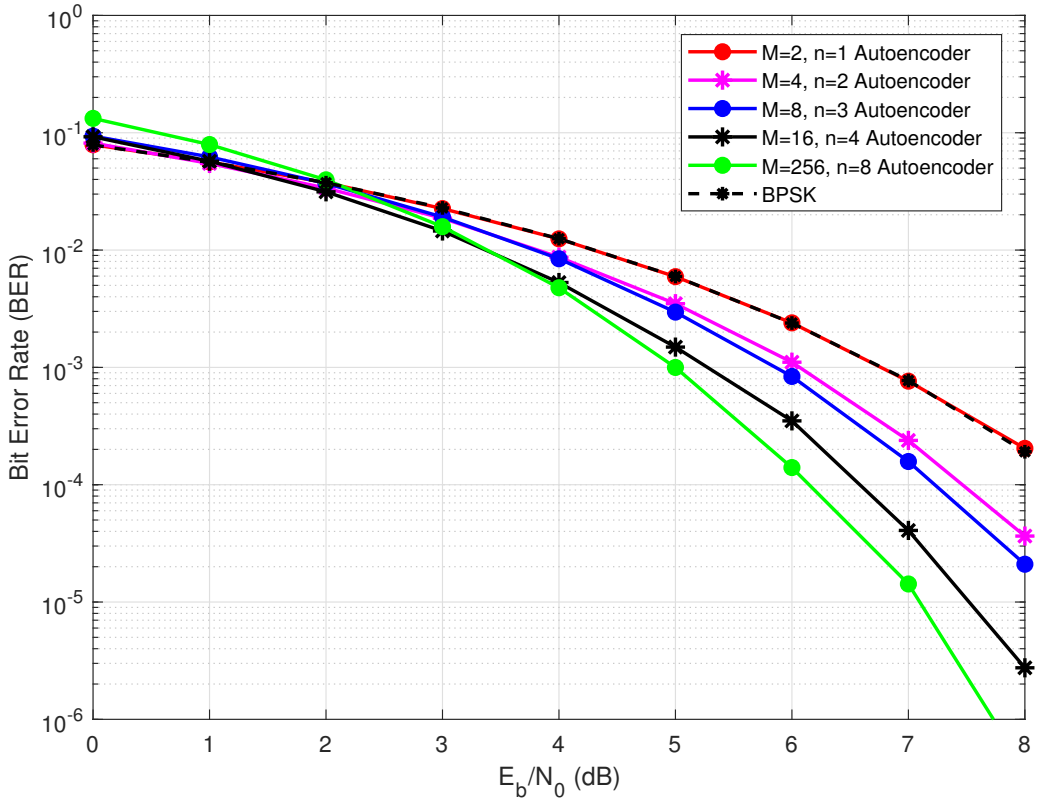

Figure 3.3. BER performance of the $R = 1$ bit/channel use systems compared with theoretical AWGN BPSK performance.

However, it should be noted that this kind of a system has a certain delay when detecting and decoding received symbols at the receiver. That is, for a system with $M = 16$ and $n = 4$, even though the communication rate of 1 bit/channel use gives the idea that we can decode 1 bit per each transmission at the receiver, we have to wait for four signalling instances to receive the complete message which consists of four symbols and then decode it so that it gives the 4 bits long message. While increasing M and n enables to have a lower BER for a system with a given communication rate, this delay in detection also needs to be taken into consideration when deciding M and n parameters.

Figure 3.4 shows the BER performance of $R = 2$ systems compared with theoretical AWGN BER performance of their baseline QPSK scheme and there also we can observe that autoencoder has better BER performance than QPSK in higher E_b/N_0 values. However, it is seen that QPSK is better when E_b/N_0 is in the low range between 0-5 dB.

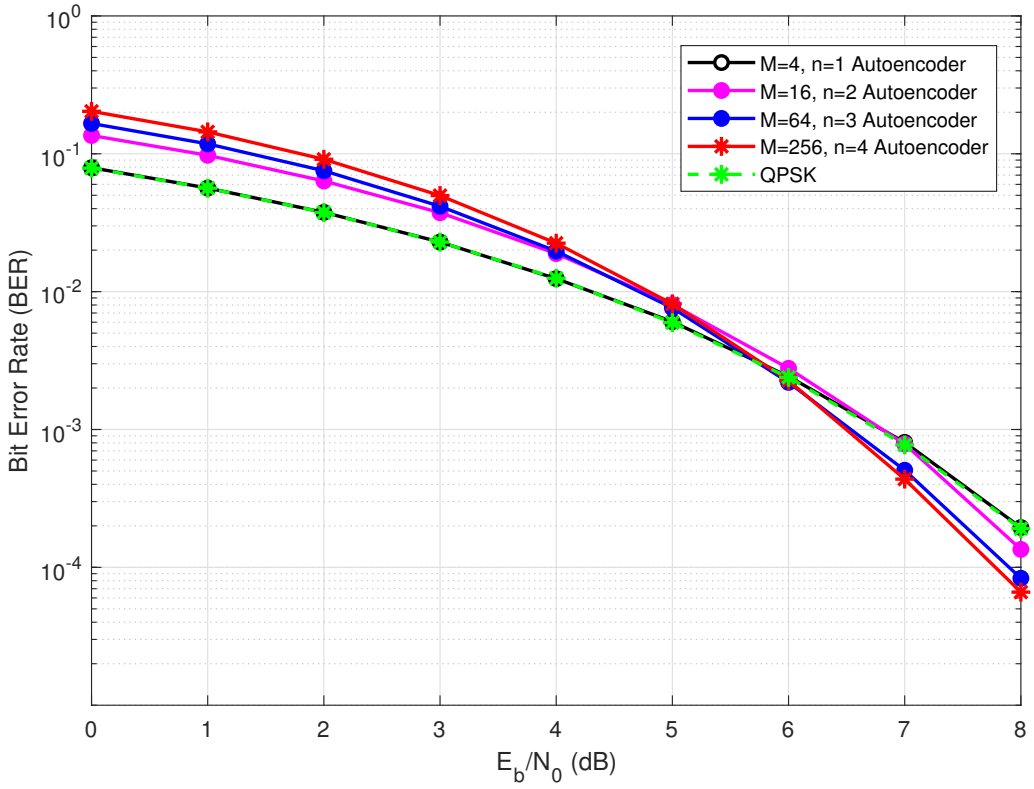

Figure 3.4. BER performance of the $R = 2$ bits/channel use systems compared with theoretical AWGN QPSK performance.

Learned Constellations

The autoencoder can be split to two parts: encoder and decoder, after training the model in end-to-end manner. Then the encoder part is implemented at the transmitter side which generates encoded symbols for each message to be sent over the channel and decoder part is implemented at the receiver which regenerates the messages from the received symbols. After completion of model training, encoder can generate all possible output signals for each message in the message alphabet. Figure 3.5 and Figure 3.6 show learned constellations for different systems we tested. When mapping $2n$-dimensional output from the encoder model to the n-dimensional complex valued vector $\mathbf{x}$, the odd indexed elements of $\mathbf{x}$ are taken as in-phase (I) components and even elements of $\mathbf{x}$ are taken as quadrature (Q) components. In the scatter plots, I and Q values are plotted in x- and y- axes respectively.

From the scatter plots in Figure 3.5, it can be seen that for the systems with $M = 1, 2$ and 4 where $n = 1$, learned constellations are similar to BPSK, QPSK and 16-PSK constellations respectively, with some arbitrary rotations. For the $M = 4$, $n = 2$ system shown in Figure 3.6, we can observe that the model has learned unique constellation points for four messages in two symbols in order to minimize the symbol estimation error at the receiver. It can be seen that in the first symbol, points marked in "$*$" have a maximum amplitude contrast in their in-phase amplitude values having high positive and negative values, and in the second symbol their signal points are closely located to each

other near zero. On the other hand, points marked in "△" have low amplitude values in the first symbol and high amplitude values in the second symbol. This arrangement has enabled the system to have a better a tolerance to channel distortions and has resulted in less symbol estimation errors at the receiver.

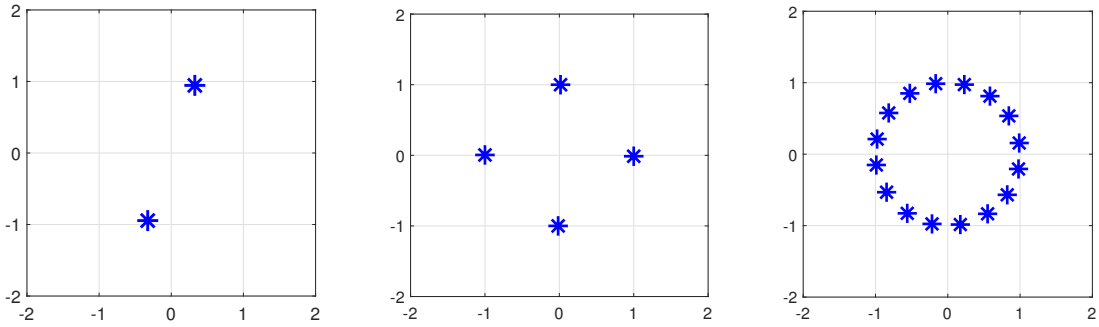

Figure 3.5. Scatter plots of learned constellations of $[M = 2, n = 1]$, $[M = 4, n = 1]$ and $[M = 16, n = 1]$ systems respectively.

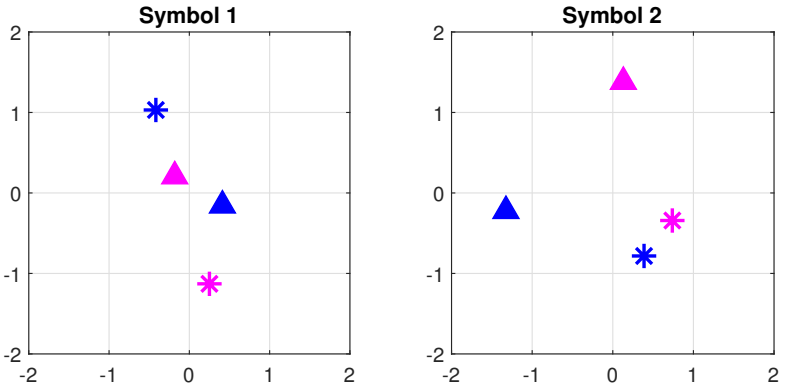

Figure 3.6. Scatter plots of learned constellations for $M = 4$, $n = 2$ system. 4 messages are shown using 4 different markers in the plot.

Effect of Training E_b/N_0 Values and Hyperparameter Tuning

During the model training it was observed that the deep model architecture and the model training parameters such as batch size and learning rate are critical factors for the performance of trained model. We have studied the BER performance of different model architectures which have single, double and multiple hidden layers in encoder and decoder parts of the autoencoder model, and it was observed that having multiple hidden layers improves accuracy than having a single hidden layer. This increased dimension parameter search space may help avoiding model convergence to sub-optimal minima during optimization as also pointed out by [5].

As noted in the previous studies also, the E_b/N_0 value in which the model training should be done plays a critical role in determining the performance of the trained model. We have tried different approaches to select training E_b/N_0 values to see the impact of them for the BER performance of the trained autoencoders. BER performance for those

approaches is shown in Figure 3.7. For this, we have used $M = 16$, $n = 4$, $R = 1$ system configuration which is equivalent to BPSK.

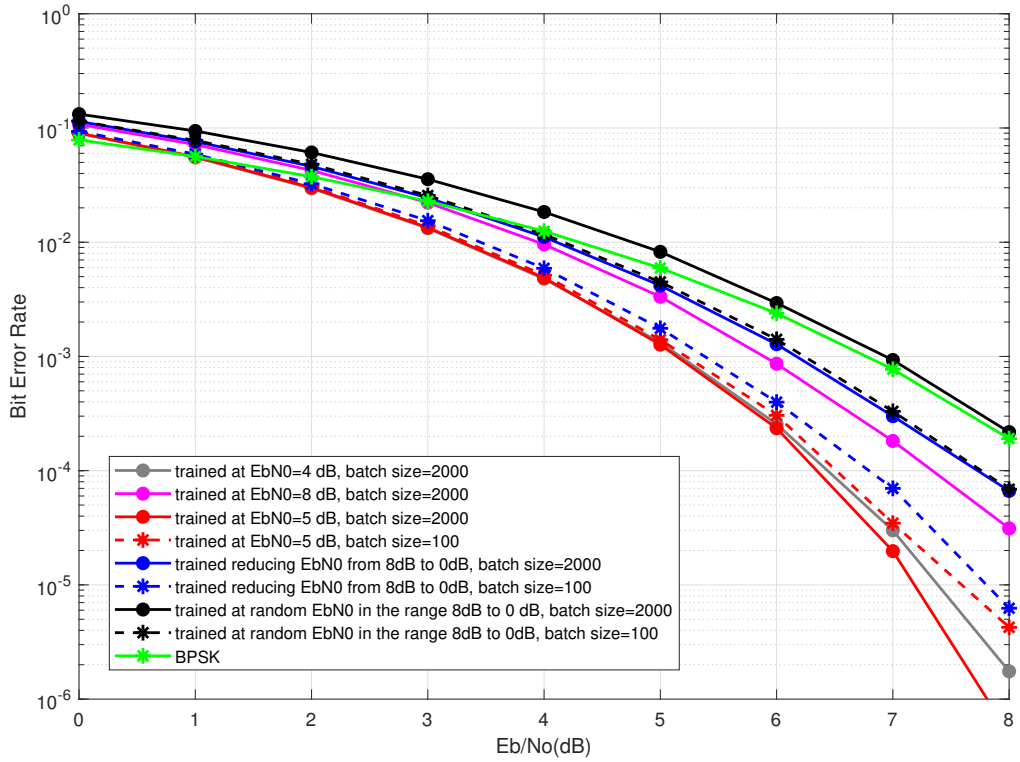

Figure 3.7. BER performance for different training E_b/N_0 values and different batch sizes. $M = 16, n = 4$ ($R = 1$ bit/channel use) system used.

From the results, we could see that training at a fixed E_b/N_0 value of 5 dB gave the best BER performance. Increasing the training E_b/N_0 seemed not that optimum as the trained model was unable to perform well in low E_b/N_0 values. When the training E_b/N_0 was selected to be too low, BER performance again degraded as the model seemed unable to capture the actual underlying patterns between inputs and outputs during training.

Different batch sizes were tried when training the models and it was observed that when training at a fixed $E_b/N_0 = 5$ dB, a larger batch size of 2000 resulted in improved BER performance compared to smaller batch sizes, while when training was done decreasing E_b/N_0 values along the training epochs, a smaller batch size around 50 or 100 gave better BER performance than higher batch sizes. Overall, training at a fixed $E_b/N_0 = 5$ dB with batch size = 2000 gave the best BER performance among the different configurations we tried.

4 END-TO-END LEARNING OF CODED SYSTEMS

In this chapter, we investigate the performance of autoencoder based end-to-end communications in comparison with conventional communications systems with channel coding applied. A conventional communications system consists of different blocks for channel coding/decoding and modulation/demodulation. Autoencoder based system does not have such explicit blocks, instead it tries to optimize the system in an end-to-end manner adhering to the configured system parameters such as input message size, number of channel uses per message and transmit signal power constraints. We use these system parameters to implement autoencoder models which are equivalent to conventional channel coded communications systems and compare their performance over the AWGN channel.

This chapter consists of two main sections. In Section 4.2, we use the same autoencoder layout used in Chapter 3 to implement systems equivalent to channel coded systems with BPSK modulation and compare the performance of autoencoder based systems and their equivalent conventional implementations. In Section 4.3, we propose a different autoencoder layout to enable implementing equivalent systems of channel coded systems with higher order modulations such as QPSK, 16-QAM etc. and compare performance of autoencoder based systems with their equivalent conventional implementations. We have mainly used BER metric to compare the performances.

4.1 System Model

As shown in Figure 4.1, in a standard communications system, the information blocks are divided to blocks for ease of processing through different stages of the system. At the transmitter, information block of size K bits is fed to channel encoder where a block of size N is output after channel encoding, and the coding rate is defined as $R = K/N$.

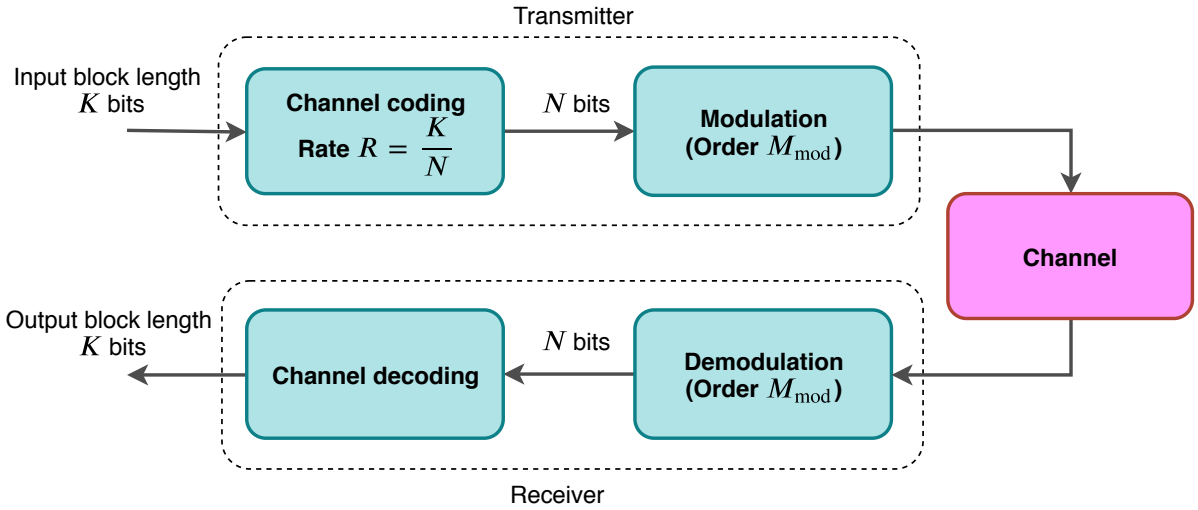

Figure 4.1. System model of a conventional communications system consisting of coding and modulation blocks.

Then the data block is fed to the modulator of order M_{mod} where the data bits are divided into codewords of size $k_{mod} = log_2(M_{mod})$, and each codeword is mapped to a point in the signal constellation with given amplitudes for the I,Q signals. At the receiver, the reverse process of the above happens where incoming symbols are mapped to codewords and the codewords are grouped serially to produce the block. It is then fed to the channel decoder where the N bit sized block is converted to K bit sized block after channel decoding which is the estimation of the transmitted information block.

In Section 4.2 and 4.3, the autoencoder model dimensions are selected in such a way that the above described network structure is preserved so that baseline system and autoencoder system can be compared.

4.1.1 Baseline for Comparison

For the channel encoding/decoding blocks in the standard communications system, we have used convolutional codes (CC) with Viterbi decoder, with hard decision decoding and soft decision decoding as the primary baseline comparison. CC with constrain length 7 is used when comparing the performance. For baseline systems, BER performance for different input block lengths with $K = \{200, 400, 800, 1600, 2000, 4000\}$ bits are evaluated to understand the effect of different block sizes for the conventional algorithms and to compare them with the autoencoder performance. Standard modulation schemes such as BPSK, QPSK, 16-QAM etc. are used in the modulation block. Conventional system was implemented in MATLAB [37] using the inbuilt functions and algorithms for coding, modulation and decoding functions and results obtained for the autoencoder models were compared with the obtained baseline results.

4.2 Autoencoder Implementation for Coded Communications Systems with BPSK Modulation for AWGN Channel

The same autoencoder model developed in Section 3.2 is used to compare the autoencoder model BER performance with the conventional coded system BER performance. Different rates R were achieved by changing ratio of the number of input bits k and number of n channel uses in the autoencoder models accordingly, so that the models resulted in equivalent systems to conventional systems with code rates $R = \{1/2, 1/3, 1/4\}$, and with BPSK modulation ($M_{mod} = 2$). The AWGN channel noise variance is given by $\beta = (2RE_b/N_0)^{-1}$. This section presents the implementation and the results obtained for the autoencoder based transmitter-receiver system in comparison with the standard communications system with different code.

4.2.1 Implementation

Autoencoder model was implemented, trained and tested in Keras with TensorFlow as backend similar to Section 3.2 and the model training was done in end-to-end over stochastic channel model using SGD with Adam optimizer with learning rate = 0.001. Same code rate values were achieved by implementing autoencoder models for different

messages sizes $M = \{2, 4, 16, 256\}$ and setting the number of channel uses (n) accordingly. Table 4.1 shows the model parameters for different simulations that were performed. For each model, energy for transmitting a message were kept equal in the autoencoder model and in the baseline system. Each model was trained over 50 epochs and mini-batch size 2000 using a training set of 1,000,000 randomly generated messages. For model training, $E_b/N_0 = 5$ dB was used. Testing the trained models were performed with 1,000,000 different messages over 0 dB to 10 dB E_b/N_0 range comparing the BER performance with their corresponding baseline system.

Table 4.1. System parameters for autoencoder models and baseline systems

Autoencoder configurations			Baseline system parameters	
Message size M	$k = log_2(M)$	Channel uses n	Code rate R	Block size K
2	1	2		200
4	2	4	1/2	
16	4	8		400
256	8	16		
2	1	3		800
4	2	6	1/3	
16	4	12		1600
256	8	24		
2	1	4		2000
4	2	8	1/4	
16	4	16		4000
256	8	32		

4.2.2 Results and Analysis

BER Performance

Figures 4.2 - 4.4 show simulated BER performances of different autoencoder models with $R = \{1/2, 1/3, 1/4\}$ and their baseline systems with convolutional coding with respective code rates and BPSK modulation scheme. Selected block length for baseline system is $K = 800$ and the constraint length of the convolutional encoder/decoder is taken as 7.

It can be observed that the BER performance of the autoencoder improves when message size is increasing. For a given code rate, $M = 2$ model has almost same performance with uncoded BPSK while $M = 256$ model has resulted in a much improved BER performance closer to the baseline. This improvement is achieved since the model has more degrees of freedom and more flexibility for a better end-to-end optimization when the message size is high.

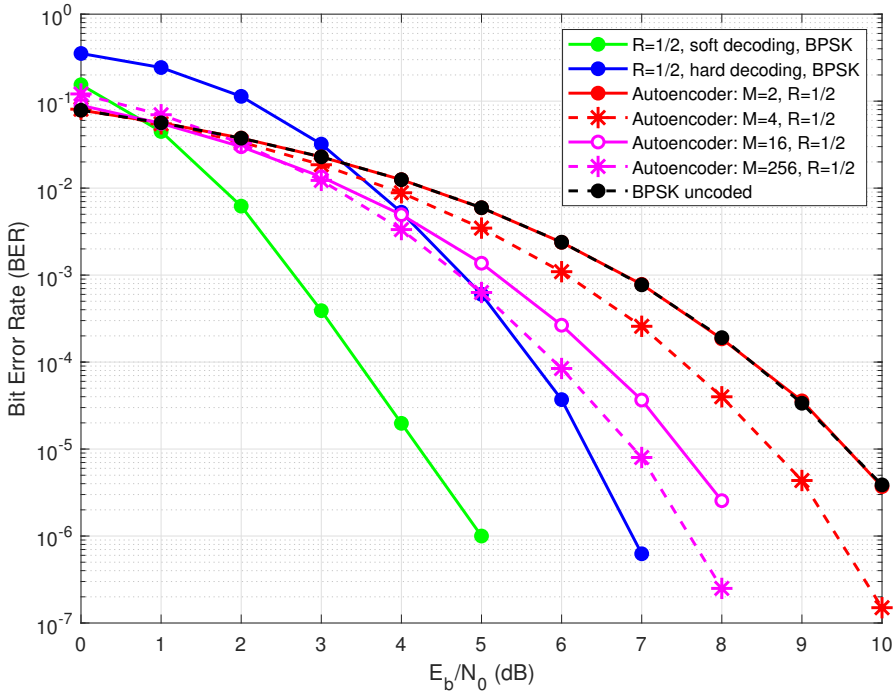

Figure 4.2. $R = 1/2$ system BER performance comparison of different autoencoder models with $M = \{2, 4, 16, 256\}$.

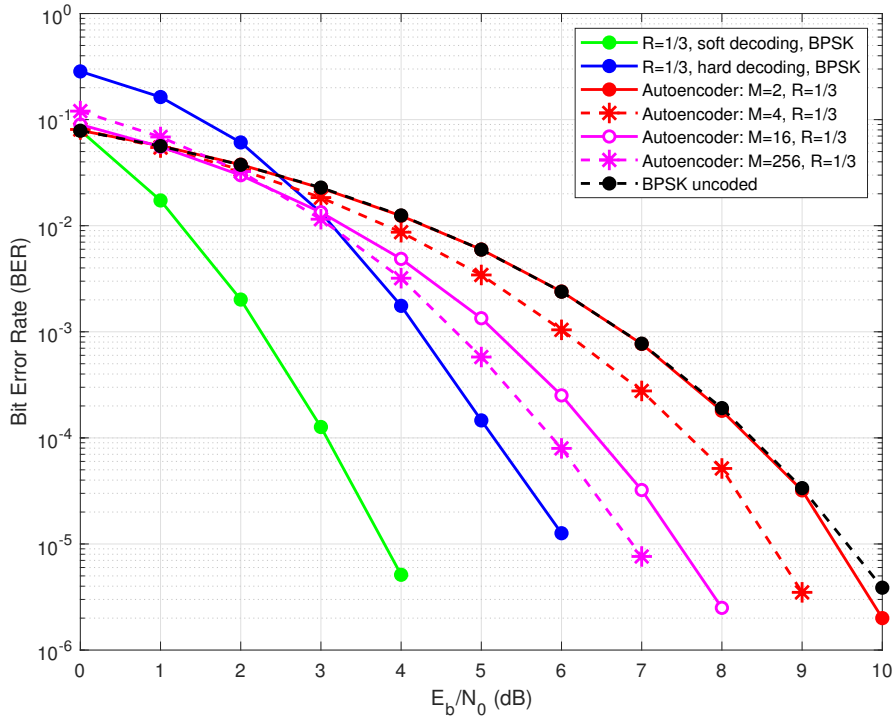

Figure 4.3. $R = 1/3$ system BER performance comparison of different autoencoder models with $M = \{2, 4, 16, 256\}$.

When considering $M = 2$ and rate $R = 1/2$ system, number of bits per message is $k = log_2(2) = 1$ and, $n = 2$ number of channel uses are there to transmit the 1 bit message. For this setup, the best possible signal formulation in each channel use would be to maximize the distance between two message constellation points which is similar to BPSK modulation. Since the autoencoder model dimensions are determined by the parameters M and R, low M values do not result in much coding gain as the non-linearities added by the model during the learning process are limited by the layer dimensions. Increasing the message size increases the layer dimensions and, for a same rate R, the model has more degrees of freedom in terms of learnable parameters which can be optimized to minimize the end-to-end message transmission error. For example, for $M = 256, R = 1/2$ system, 16 channel uses are there to transmit 256 different messages which has more flexibility than the earlier scenario where 2 messages are transmitted in 2 channel uses. Thus, when $M = 256, R = 1/2$, the model has been able to learn the transmit symbols with a channel coding gain as expected, which can be observed from the BER plots. Table 4.2 compares the number of learnable parameters in each layer in above two scenarios which helps us understanding how the model learning capacity increases with increasing message size.

Table 4.2. Learnable parameters $R = 1/2$ autoencoder models

Layer (Output dimensions)	Number of parameters		
	(M, n)	(M=2, n=2)	(M=256, n=16)
Input (M)	0	0	0
Dense-ReLU (M)	(M+1)*M	6	65792
Dense-ReLU (M)	(M+1)*M	6	65792
Dense-Linear (2n)	(M+1)*2n	12	8224
Normalization (2n)	0	0	0
Noise (2n)	0	0	0
Dense-ReLU (M)	(2n+1)*M	10	8448
Dense-ReLU (M)	(M+1)*M	6	65792
Dense-Softmax (M)	(M+1)*M	6	65792

Even though the autoencoder BER performance is always worse than soft decision CC, it can be observed that autoencoder has a comparable performance to the hard decision CC, specially when the code rate is high. For $R = 1/2$, autoencoder with $M = 256$ is better than hard decision CC in low E_b/N_0 range from 0 dB to 5 dB and it is only around 1 dB worse than the hard decision CC at a BER of 10^{-5}.

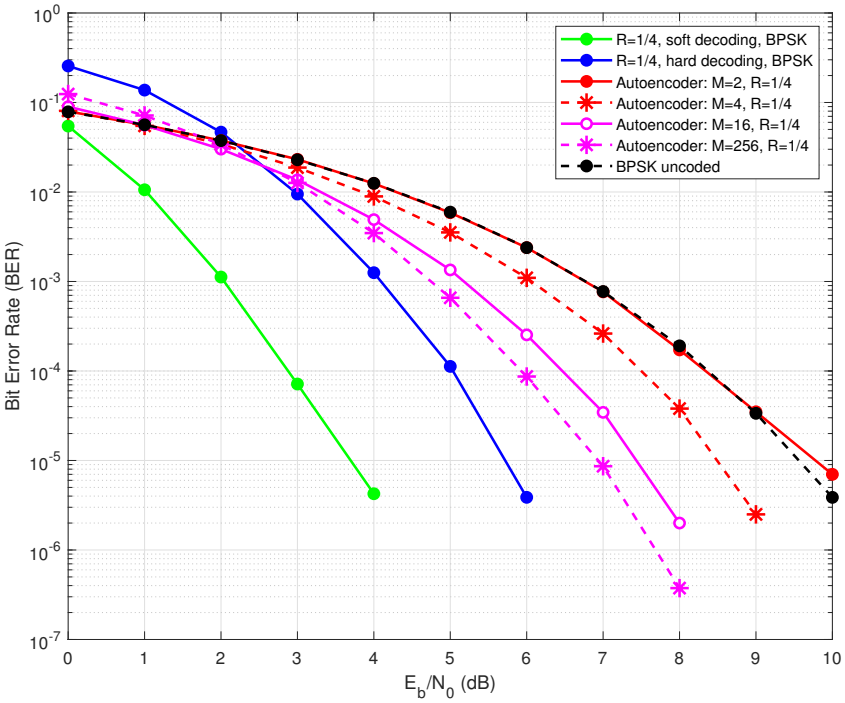

Figure 4.4. $R = 1/4$ system BER performance comparison of different autoencoder models with $M = \{2, 4, 16, 256\}$.

Figure 4.5 shows the message error rate performances of different autoencoder models with $R = \{1/2, 1/3, 1/4\}$ and $M = 256$. We can observe that for the same message size, three models with different rates have resulted in almost same MER, and the models have an acceptable MER performance with a 10^{-5} error at 7 dB.

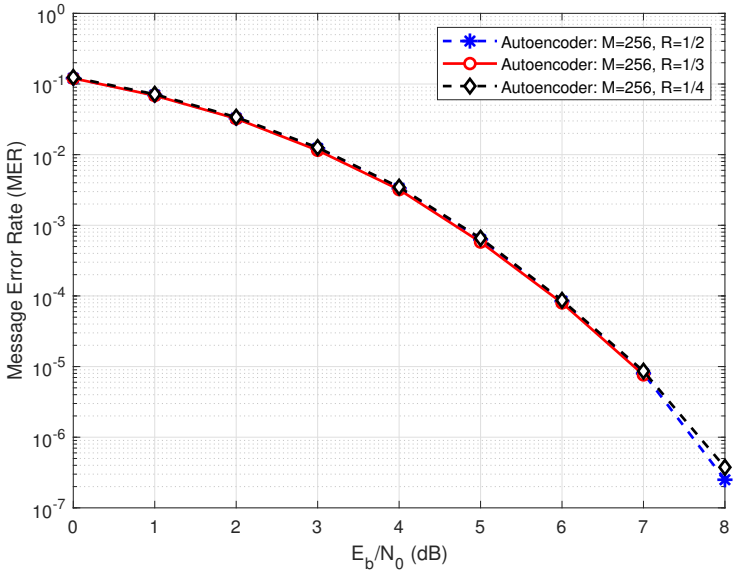

Figure 4.5. MER performance of different autoencoder models with $R = \{1/2, 1/3, 1/4\}$ and $M = 256$.

Figures 4.6 and 4.7 compare the autoencoder BER performance for $R = \{1/2, 1/4\}$ when different block lengths are used in the baseline system with hard decision CC. It can be observed that for $R = 1/2$ system, baseline BER performance is better for mid-range block lengths where K is 400 - 800 bits. The performance degrades when block size is very low as 200 bits or high as 2000 - 4000 bits, reducing the gap between the autoencoder and baseline BER performance. For $R = 1/4$ hard decision CC, there is not much effect on block size for the BER. On the other hand, autoencoder performance is independent of block length K as for a given model with message size M, its input size is $k = log_2(M)$ bits where the K bit long block is divided to k bit long sub-blocks and fed to the system.

Thus, from the results we obtained, autoencoder models would be more effective to be used in high or low input block length scenarios for systems with higher code rates where they have comparable performance to the baseline.

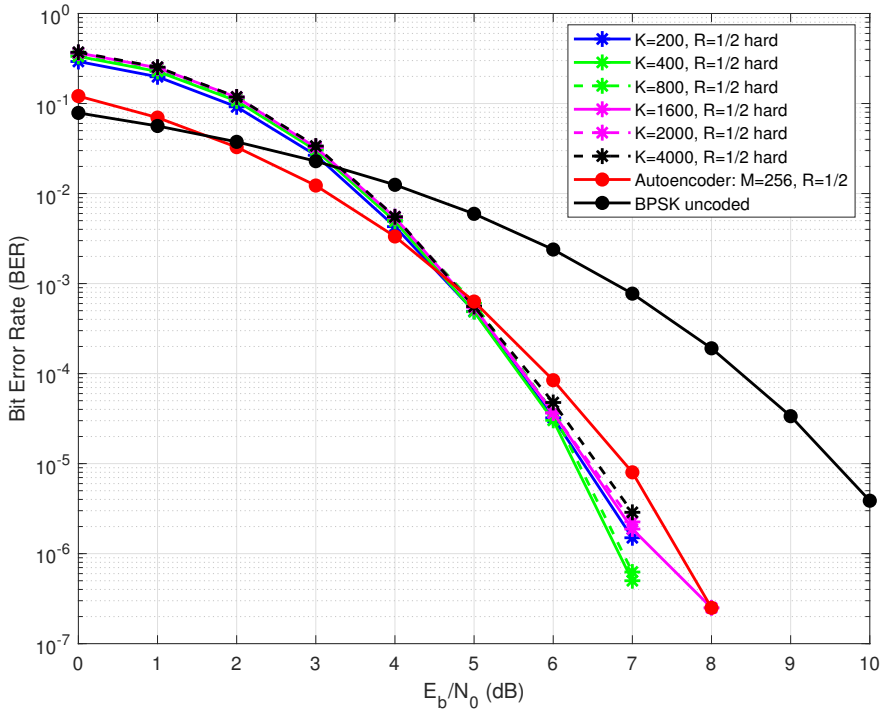

Figure 4.6. $R = 1/2$ system BER performance comparison for different block lengths: $K = \{200, 400, 800, 1600, 2000, 4000\}$.

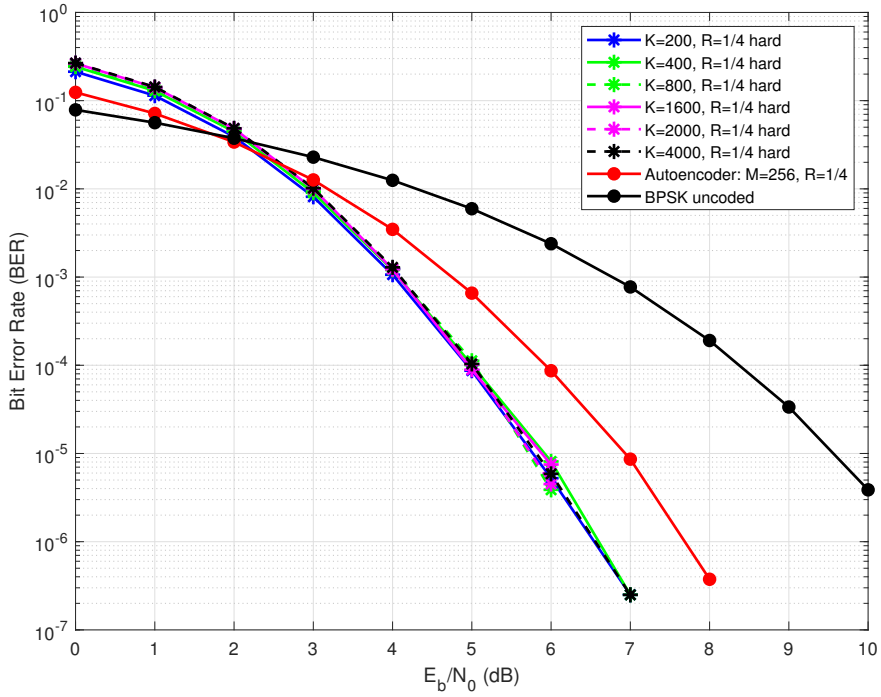

Figure 4.7. $R = 1/4$ system BER performance comparison for different block lengths: $K = \{200, 400, 800, 1600, 2000, 4000\}$

Learned Constellations

Figures 4.8 and 4.9 show the learnt constellations for different systems we tested for a same code rate of $R = 1/2$. Same as in Chapter 3, when mapping $2n$-dimensional output from the encoder model to the n-dimensional complex valued vector $\mathbf{x}$, the odd indexed elements and even indexed elements of $\mathbf{x}$ are taken as I and Q components respectively. In the scatter plots, I and Q values are plotted in x- and y- axes respectively. $M = 4, R = 1/2$ system uses 4 symbols to transmit a single message and Figure 4.8 shows the signal points in all 4 symbols. It can be observed that the model has learned unique constellation points for 4 messages in four symbols in order to minimize the symbol estimation error at the receiver. $M = 16, R = 1/2$ system uses 8 symbols to transmit a single message and the learned signal points for each of the 8 symbols are shown in Figure 4.9.

From the constellation diagrams we can observe that the autoencoder system does not have a fixed constellation as in equivalent BPSK modulation scheme. In a conventional communications system, output from the channel coding block is binary valued and each bit in the coded block is mapped to the BPSK constellation accordingly. Thus, the signal transmission in each channel use is independent of others and each signal carries independent bit information. In contrast, in the autoencoder, the transmit signals are temporally correlated to each other as n signals in n channel uses transmit the message s a whole. In the autoencoder implementation, the model uses the available number of channel uses (or number of symbols) per transmit message and learns the optimum I, Q signal values for each channel use to transmit the message with a minimum reconstruction error at the receiver. This approach results in learning a joint coding and modulation scheme utilizing the available channel uses to transmit a given message depending on the

system parameters (number of input bits per message and number of channel uses per message) in order to achieve the maximum possible tolerance to the distortions caused by the noise added in the channel. Thus, even for a same rate R ($R = 1/2$ in this case), changing the message size results in different constellations as the model dimensions and the learned parameters are different for different message sizes.

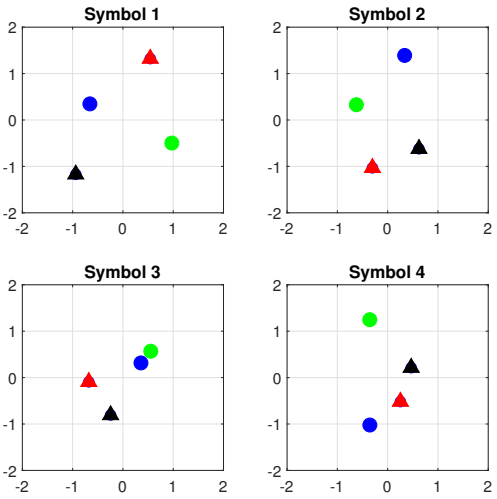

Figure 4.8. Scatter plots of learned constellations for $M = 4, R = 1/2$ system. 4 messages are shown using 4 different markers in the plot.

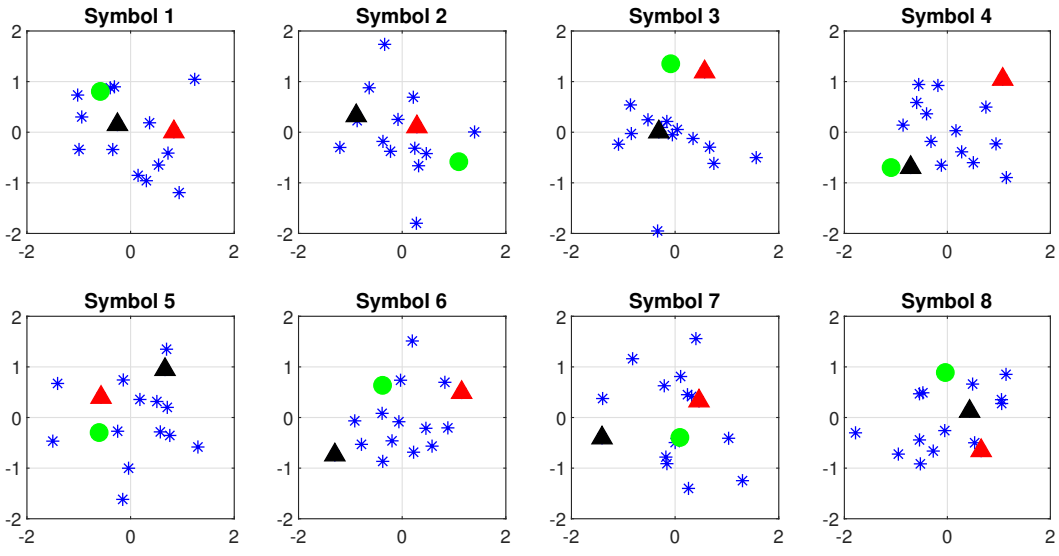

Figure 4.9. Scatter plots of learned constellations for $M = 16, R = 1/2$ system. 3 different messages are shown using 3 different markers in the plot.

4.3 Autoencoder Implementation for Coded Communications Systems with Higher Order Modulations

A new autoencoder architecture capable of implementing a channel coded communications system configuration with different modulation schemes is introduced in this section and its performance in comparison to conventional system is presented. For that, we have improved the original autoencoder layout proposed by [5], changing the dimensions of the layers of the model in order to absorb different parameter settings in a conventional communications system.

4.3.1 Implementation

The general setup of a typical conventional communications system described in Section 4.1 was incorporated into an autoencoder layout as depicted in Table 4.3. The internal layers of the proposed autoencoder architecture are designed with parameters relating to the channel encoder, modulator functions at the transmitter side and demodulator, channel decoder functions at the receiver side.

In contrast to the original autoencoder architecture where input was an M-dimensional *one-hot encoded* vector to indicate a message s out of M possible messages, in this autoencoder the input message s is given as a binary vector of k bits where $k = log_2(M)$. Let $n_{coded} = k/R$ where n_{coded} is the number of coded bits per each message of k bits. Then, for a given modulation order M_{mod} with codeword size of $k_{mod} = log_2(M_{mod})$, $n = n_{coded}/k_{mod}$ symbols are required to transmit the encoded n_{coded} bits. The dimensions of each layer of the new autoencoder architecture are determined using these defined parameters. When comparing with a conventional system, the N bit information block is to be divided into k bit long messages when feeding into the autoencoder. The output of the decoder is also k bits long, which gets mapped to the estimated message $\hat{s}$. Figure 4.10 illustrates the layout of the autoencoder model. Layer types, activation functions and dimensions of each layer in the model are listed in Table 4.3.

Table 4.3. Layout of the autoencoder model equivalent to coded systems with higher order modulations

Layer	Output dimensions
Input	k
Dense-ReLU	n_{coded}
Dense-ReLU	$2n$
Dense-ReLU	$2n$
Dense-Linear	$2n$
Normalization	$2n$
Noise	$2n$
Dense-ReLU	$2n$
Dense-Linear	n_{coded}
Dense-Sigmoid	k

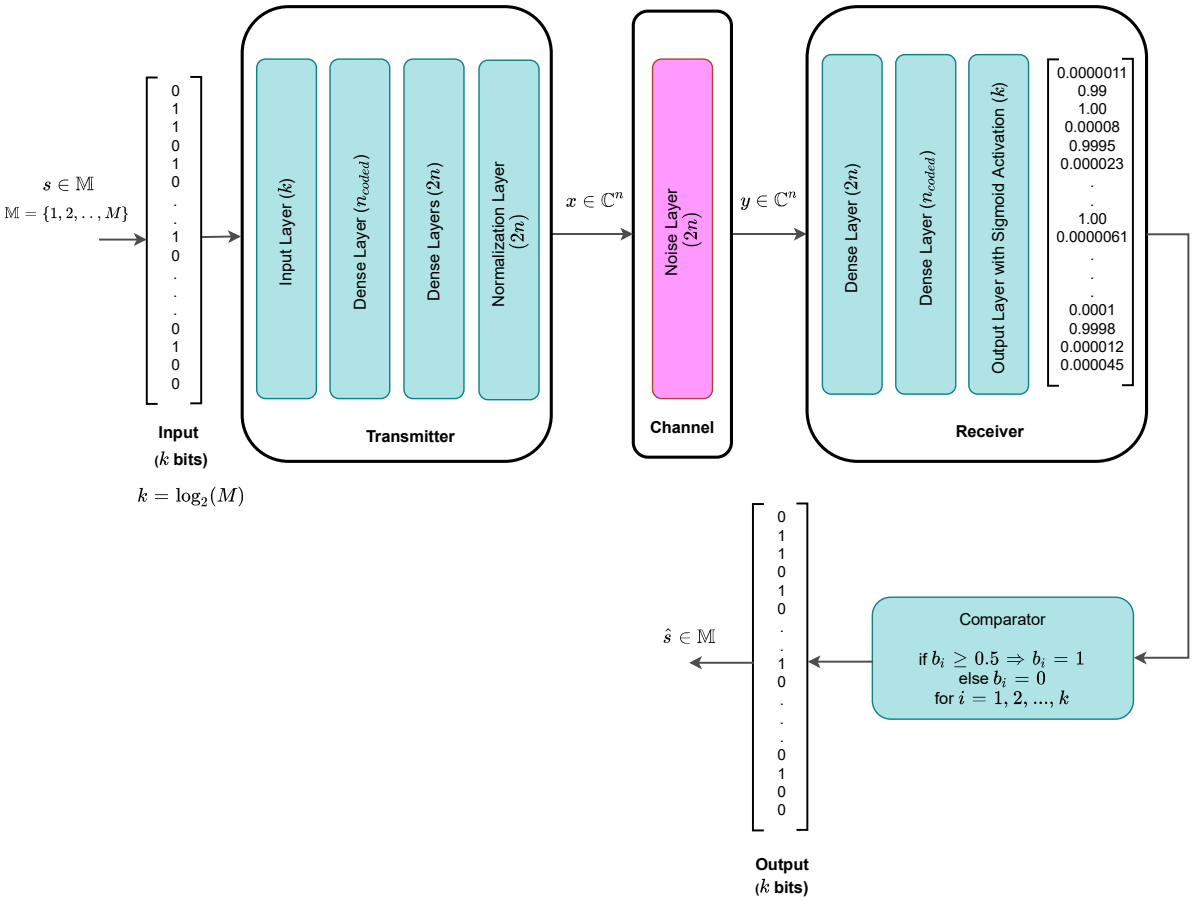

Figure 4.10. Autoencoder model implementation equivalent to coded systems with higher order modulations.

Since the input to the model is a binary vector and we expect a reconstruction of the input vector at the output of the autoencoder, it is essential to have an output layer which gives values $0s$ and $1s$ as output. Thus, we have implemented a fully connected layer with the *Sigmoid* activation function at the output layer (which has outputs in the range $(0, 1)$) along with the binary cross-entropy loss function for model training which results in each of the k bits of the output vector to be closer to either 1 or 0 after end-to-end optimization of the model to minimize the loss. After the model training, the autoencoder output can be applied to a simple comparator module to produce the binary outputs as shown in Figure 4.10.

Autoencoder is trained end-to-end over the stochastic channel model using SGD method with the Adam optimizer with learning rate = 0.001. Same as in the earlier simulations, model training and testing was implemented in Keras with TensorFlow. Different models were trained for different message sizes ($M = 16, 64, 256, 4096$ etc.), code rates and modulation schemes such as QPSK, 16-QAM etc. The AWGN channel noise variance is given by $\beta = (2Rk_{mod}E_b/N_0)^{-1}$ and for model training, the channel is represented by an additive noise layer with fixed variance β. Each model was trained over 100 epochs and with batch size = 1000 with a training set of 1,000,000 randomly generated messages. $E_b/N_0 = 5$ dB was used for model training. Testing the trained models were performed with 1,000,000 different random messages over 0 dB to 10 dB E_b/N_0 range and their BER performances have been compared with the corresponding

baseline systems. Table 4.4 below summarises the simulation parameters on which we have tested the models.

Table 4.4. System parameters for autoencoder models and baseline systems

Autoencoder configurations				Baseline system parameters	
M	$k = log_2(M)$	n_{coded}	n	R	M_{mod}
16	4	8	4	1/2	4
64	6	12	6		(QPSK)
256	8	16	8		
256	8	16	4	1/2	16
4096	12	24	6		(16-QAM)

4.3.2 Effect of Model Layout and Hyperparameter Tuning for the Performance

Determining the autoencoder architecture including the number of layers, layer dimensions, activation functions for the layers and the cost function plays a critical role in achieving a better performance. The autoencoder structure proposed in Section 4.3.1 was selected considering all those aspects. It was observed from the results in Chapter 3 that having multiple hidden layers improves accuracy than having a single hidden layer, due to the increased dimension parameter search space which results in a better optimization avoiding the model convergence to sub-optimal minima. Based on that observation, and also considering the conventional communications system parameters at each block of the communications system, the above mentioned autoencoder model was selected to optimally incorporate those system parameters into a multi-layered autoencoder architecture having the layer dimensions accordingly.

In terms of determining suitable activation functions for the layers, *Linear* and *Sigmoid* activation functions were selected for the last layer of the encoder block and the last layer of the decoder block respectively, so that they are equivalent to the conventional communication system settings as explained in Section 4.3.1. For the hidden layers, different types of activation functions such as *Linear* and *ReLU* were tried out during preliminary simulations for model selection, and the activation functions which resulted in better BER performance are listed in Table 4.3 for each layer of the autoencoder.

Furthermore, during the preliminary simulations, the autoencoder performance when using binary cross-entropy loss function and mean square error loss function were both evaluated as they were suitable candidates for the proposed model structure and the model input-output configuration. For a given autoencoder configuration, binary cross-entropy and mean squared error (MSE) loss functions resulted in different model training loss and model accuracy values as expected. The reason for this is that they have different cost surfaces which result in different optimizations during the model training process. The initial results showed that binary cross-entropy loss function resulted in better model training accuracy and a better BER performance for the testing set. Therefore, it was selected as the loss function for the proposed autoencoder model.

Even though an extensive search for determining optimum batch size and number of epochs for model training was not done, batch size = 1000 and epochs = 100 were observed to give better results after performing some initial simulations with different configuration settings and hence those values were used when training the models. As results from Chapter 3 showed a better BER performance when model training was done at $E_b/N_0 = 5$ dB, same E_b/N_0 value was used for training the new autoencoder models as well.

4.3.3 Results and Analysis

BER Performance

Figure 4.11 shows BER performance comparison between the autoencoder and baseline system for $R = 1/2$ with 16-QAM modulation. Selected block length for baseline system is $K = 800$ and the constraint length of the convolutional encoder/decoder is taken as 7.

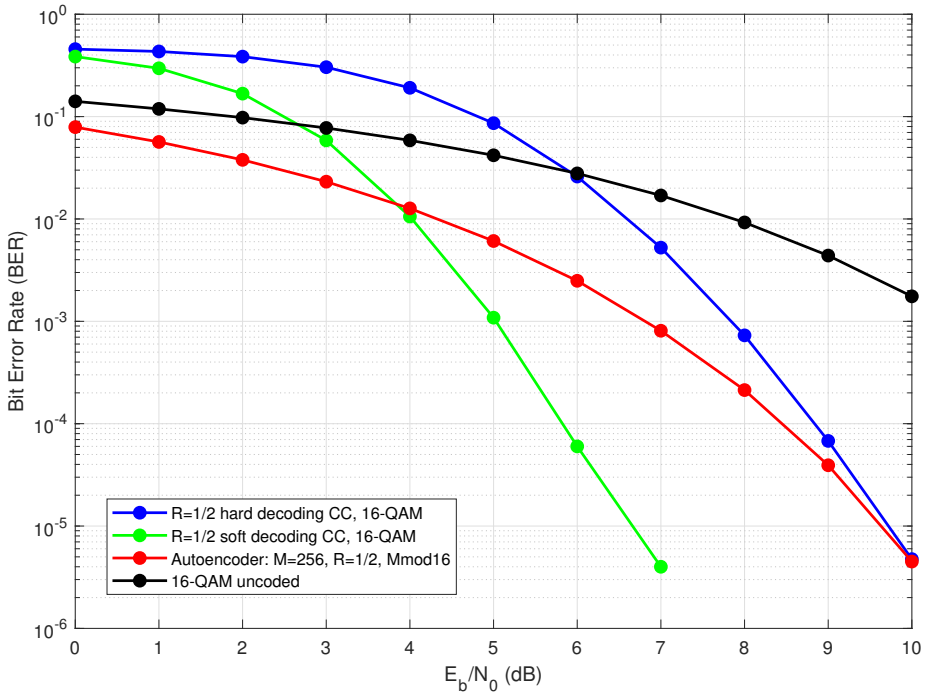

Figure 4.11. BER performance for baseline and autoencoder for $R = 1/2$, 16-QAM system.

Autoencoder BER performance is better than baseline convolutional coded system with hard decision decoding over the full E_b/N_0 considered. However, it can be noticed that the difference between BER decreases along with increasing E_b/N_0, having almost equal performance at $E_b/N_0 = 10$ dB. Baseline CC implementation with soft decision decoding is better than autoencoder for higher E_b/N_0 values. However, it can be observed that while the coded soft decision CC BER performance is worse than uncoded 16-QAM at

lower E_b/N_0 values, autoencoder has a better BER performance than soft decision CC and uncoded 16-QAM in 0 dB to 4 dB E_b/N_0 range.

BLER comparison between the autoencoder and baseline systems is shown in Figure 4.12. It can be observed that autoencoder BLER performance is worse compared to the baseline. This can be explained since the optimization criteria for the autoencoder was not the BLER, but the message error rate (MER) or the BER of each message transmitted. Figure 4.13 shows the MER performance along the considered E_b/N_0 range and we can observe that it has an acceptable MER, having less than 10^{-4} error at 10 dB.

The autoencoder based system does not require larger input block sizes to operate as the input size to the model is k bits at a time. Thus, it can achieve an acceptable BER and MER performances as shown, with only k bits ($k = 8$ in this case) which is a very low block size compared to conventional systems which typically operate with 100s or 1000s bits long block sizes. Such a system would be advantageous for low latency and low throughput communications as short message transmission can be achieved with an acceptable error performance, and with less processing complexity and processing delay than in the conventional systems.

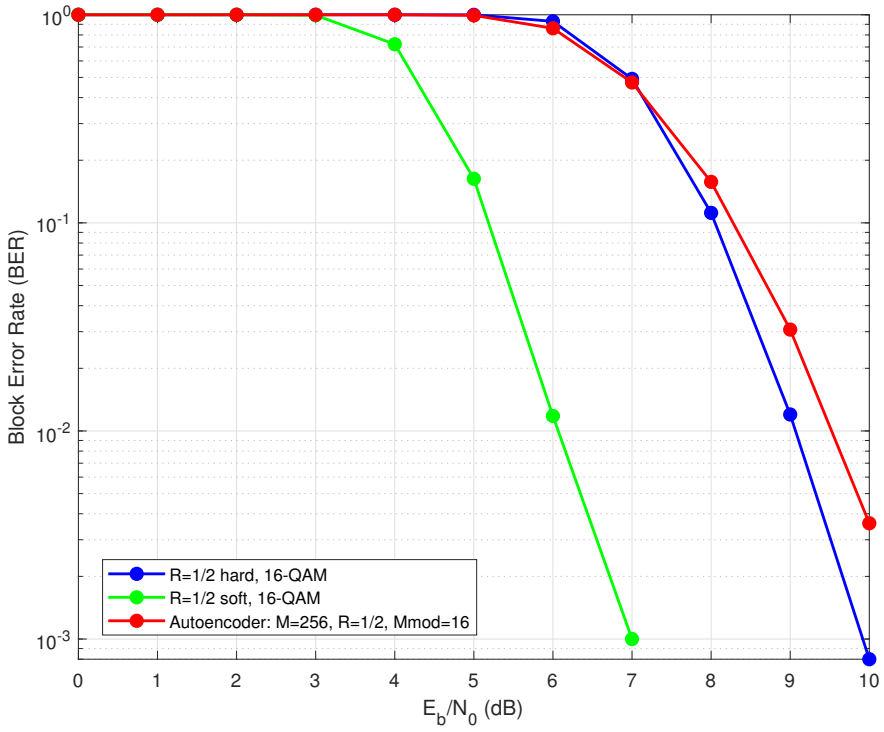

Figure 4.12. BLER performance for baseline and autoencoder for $R = 1/2$, 16-QAM system.

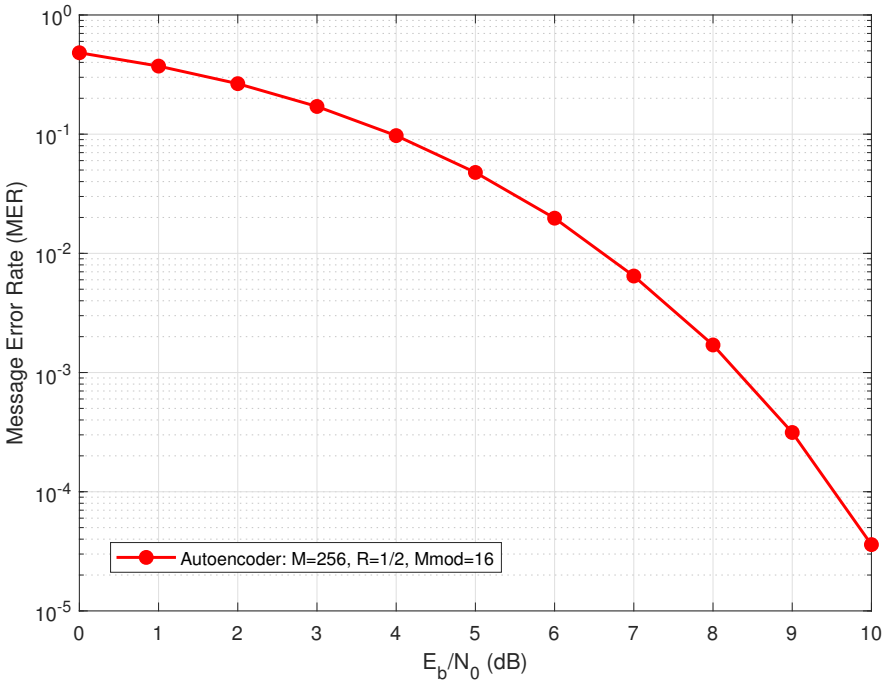

Figure 4.13. MER performance for $M = 256$, $R = 1/2$, $M_{mod} = 16$ autoencoder model.

Figures 4.14 and 4.15 compare the hard decision CC and soft decision CC performances for different block lengths respectively along with the autoencoder performance. Autoencoder performance is independent of block length K as its input size is k bits where K bit long block is divided to k bit sub-blocks and fed to the system. Autoencoder is better than hard decision CC over the full E_b/N_0 range for all the block sizes checked and is around 3 dB worse than soft decision CC at a BER of 10^{-5}.

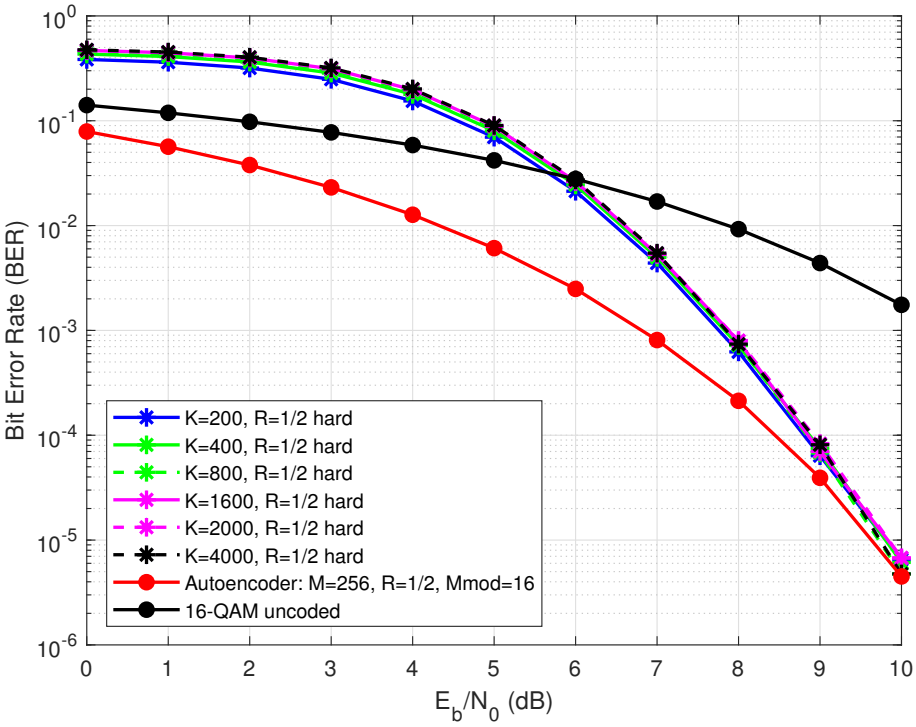

Figure 4.14. BER performance for baseline and autoencoder for $R = 1/2$, 16-QAM system with different block lengths: $K = \{200, 400, 800, 1600, 2000, 4000\}$.

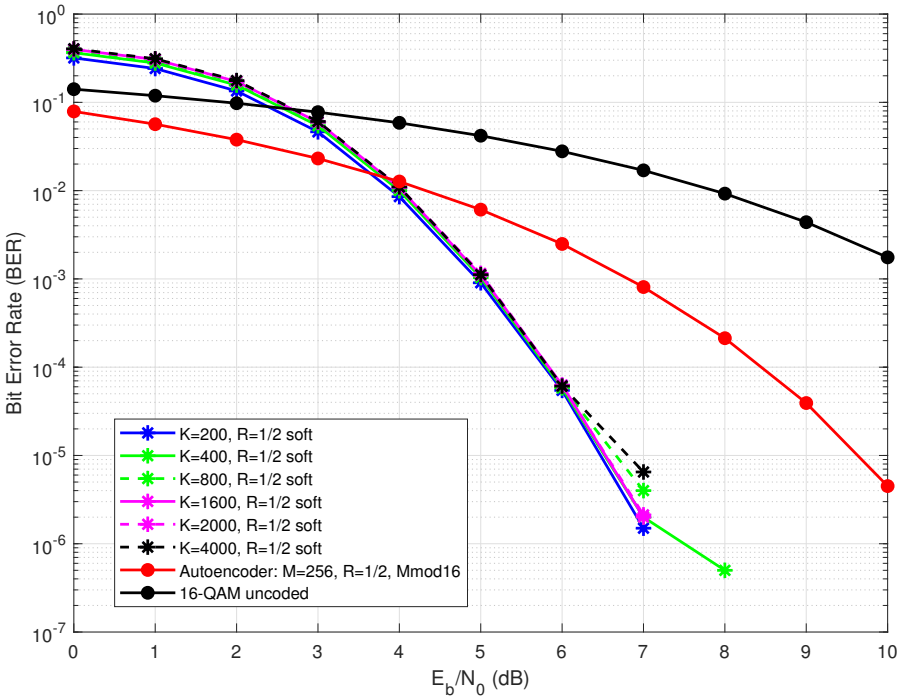

Figure 4.15. BER performance for baseline and autoencoder for $R = 1/2$, 16-QAM system with different block lengths: $K = \{200, 400, 800, 1600, 2000, 4000\}$.

Different autoencoder models were trained in order to investigate the performance of equivalent DL based systems to the conventional $R = 1/2$ QPSK system and it was observed that most of the models didn't provide the expected BER improvement. Figure 4.16 shows the achieved BER performances of several different autoencoder models with $M = \{16, 64, 256\}$ trained at $E_b/N_0 = 5$ dB. Despite having an accuracy over 99% for the trained models, testing them over the full E_b/N_0 range resulted in a BER performance similar to the uncoded QPSK, which is a somewhat disappointing observation since the autoencoder is unable to achieve the coding gain.

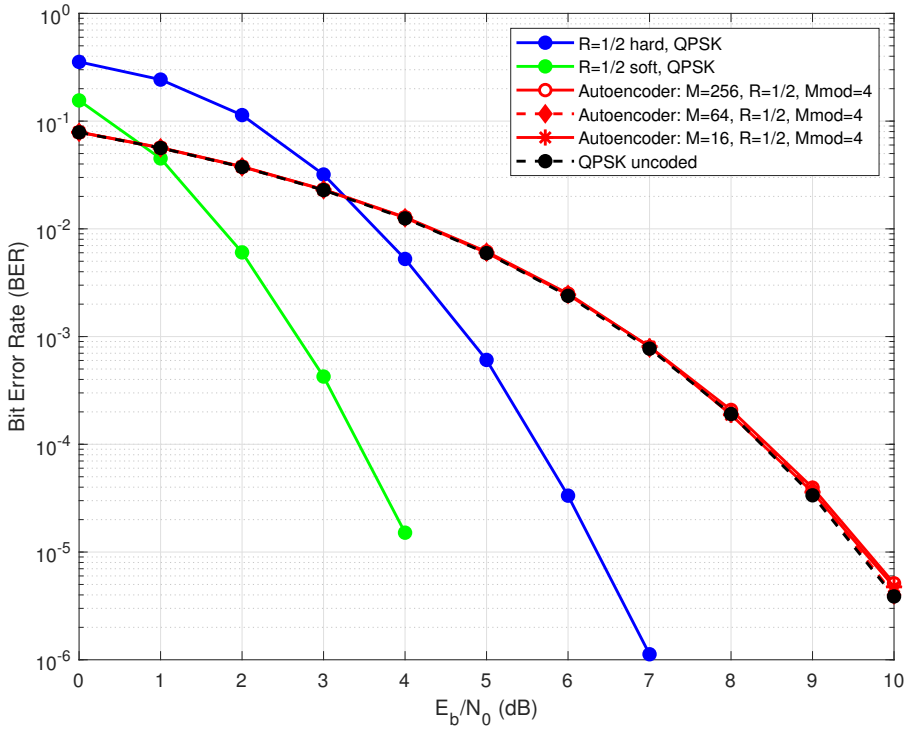

Figure 4.16. BER performance for baseline and autoencoder for $R = 1/2$, QPSK system. Autoencoder models are implemented with different message sizes $M = \{16, 64, 256\}$.

We tried training the $M = 256, R = 1/2, M_{mod} = 4$ model at different E_b/N_0 values and Figure 4.17 illustrates the BER performances of the models trained at $E_b/N_0 = \{0, 5, 8\}$ dB. It was observed that the model trained at $E_b/N_0 = 2$ dB is having much improved BER performance at low E_b/N_0 range. It is around 1.5 dB better than hard decision CC in the 0 to 4 dB E_b/N_0 range and it is only about 1 dB worse than soft decision CC in the said E_b/N_0 range. It is interesting to see that training the model at a very low E_b/N_0 value has resulted in learning optimum an transmission mechanism to overcome the high distortions caused by the channel in low E_b/N_0 range. That is, the learnt signalling strategy is more robust for low E_b/N_0 range. However, it can be seen that the learned transmission mechanism is not suitable for higher E_b/N_0 values as the performance is worse even than the uncoded case. This result shows the possibility of training a deep learning model to be optimum for a specific E_b/N_0 range. Thus, instead of having a single model with a fixed transmitter-receiver mechanism to suit the full E_b/N_0 range, it might be possible to develop multiple models with different transmitter-receiver arrangements

to suit different operating environments according to the E_b/N_0. Having the flexibility to design such multiple systems which operate under the same system parameters (i.e. same R and same modulation order M_{mod}) can be noted as an advantage of deep learning based communications systems since conventional systems generally have fixed setups.

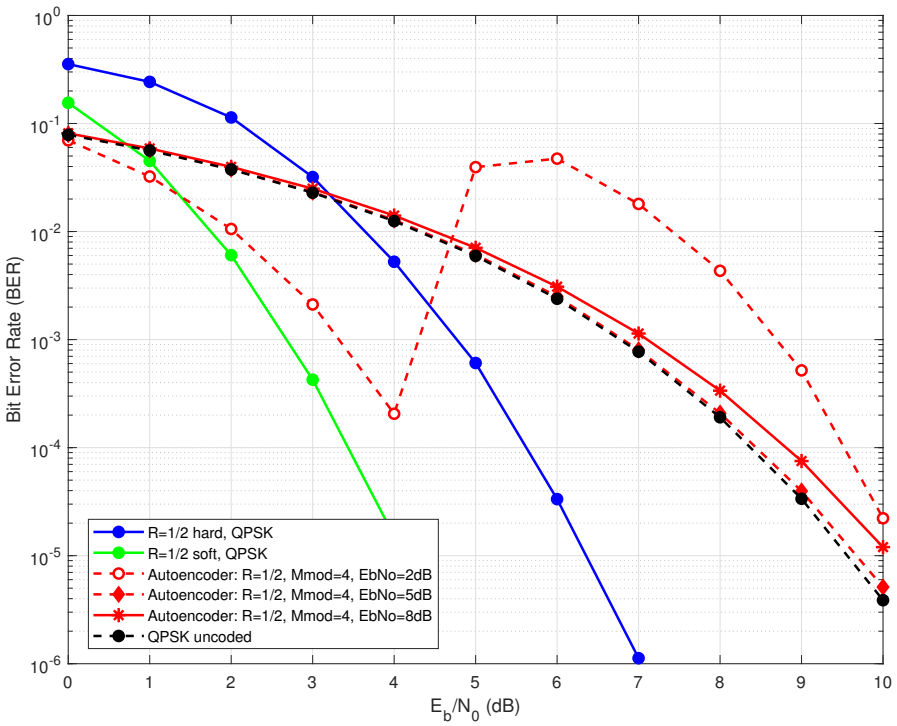

Figure 4.17. BER performance for baseline and autoencoder models for $R = 1/2$, QPSK system.

4.3.4 Processing Complexity

Processing complexity is often considered as an important parameter to determine suitable algorithms in each of the processing blocks in a conventional communications system since it is directly associated with several factors such as communications latency, power and resource requirement for implementation etc. Generally, channel coding algorithms have higher algorithmic complexity resulting in higher computational complexities associated with both encoding and decoding blocks [38]. For example, in polar codes, there can be additional complexities in the encoding and puncturing schemes. For decoding algorithms which involve iterative decoding, decoder has a higher complexity compared to the encoder. In this section we do a rough comparison of the autoencoder processing complexity with the channel coding algorithms. However, it should be noted that this is not an exact comparison since autoencoder complexity is analysed as an end-to-end system consisting of transmitter and receiver components where as in the conventional system we mainly discuss decoder complexity, but a conventional system has other blocks such as modulation/demodulation, signal detection which possess a considerable processing overhead as well.

In [38], the authors have compared the decoder complexities of different candidate channel coding algorithms for URLLC, and the Viterbi algorithm which is used in the decoder of the convolutional codes have a computational complexity of $4.R.N.2^m$ where R, N and m denote code rate, code block length and memory order respectively.

For a neural network, if there are M neurons in a hidden layer and N inputs to that layer, there are $N.M$ multiplications and M separate additions over $N+1$ terms, and M applications of the transfer function $f()$. Thus, the number of mathematical operations depend on the number of layers in the neural network and the dimensions of each layer. Table 4.5 below summarises the number of additions, multiplications and transfer function applications in each layer in the autoencoder model we implemented in Section 4.3. However, in implementation, neural networks are generally implemented with parallel processing architectures as each neuron in a layer only depends on the inputs from the previous layer, the learned parameters (or weights), and the application of the transfer function. Thus, when considering the parallel processing implementation, processing complexity in each layer for a single parallel path would be just N multiplications and $N+1$ additions and a single application of the transfer function. Viterbi decoder and autoencoder has the same range of processing complexity without considering the parallel implementation of the autoencoder. Therefore, when considering a parallel processing implementation, the autoencoder has a very low processing complexity compared to conventional system given that autoencoder caters for end-to-end processing including both transmitter and receiver side processing. Thus, the autoencoder based systems can be implemented with very low processing complexity and hence lower processing delays than the conventional systems can be achieved, which is another advantage when considering implementation of low latency systems.

Table 4.5. Number of mathematical operations in the autoencoder model

Layer	Output dimensions	Multiplications	Additions	Transfer function
Input	k	-	-	-
Dense-ReLU	n_{coded}	$k.n_{coded}$	$n_{coded}(k+1)$	n_{coded}
Dense-ReLU	$2n$	$n_{coded}.2n$	$2n(n_{coded}+1)$	$2n$
Dense-ReLU	$2n$	$2n.2n$	$2n(2n+1)$	$2n$
Dense-Linear	$2n$	$2n.2n$	$2n(2n+1)$	$2n$
Noise	$2n$	-	-	-
Dense-ReLU	$2n$	$2n.2n$	$2n(2n+1)$	$2n$
Dense-Linear	n_{coded}	$2n.n_{coded}$	$n_{coded}(2n+1)$	n_{coded}
Dense-Sigmoid	k	$n_{coded}.k$	$k(n_{coded}+1)$	k

4.3.5 Comparison with 5G Channel Coding and Modulation Schemes

The latest release (release 15) of the cellular standard in the 3rd Generation Partnership Project (3GPP) has announced the specifications for the 5G new radio (NR) air interface [39]. Compared to fourth generation (4G) long-term evolution (LTE), in 5G NR, two new channel coding techniques have been adopted, for data channels and control channels respectively. Specifically, low density parity check (LDPC) codes are to replace turbo codes used in 4G LTE for data channels and polar codes are to replace tail biting convolutional codes (TBCCs) for control channels [39]. When considering the modulation schemes, BPSK, QPSK, 16-QAM, 64-QAM and 256-QAM are adopted for 5G NR [40].

Performance of different channel coding schemes for 5G with modulation schemes such as QPSK and 16-QAM have been investigated in [41] and [42]. When comparing their results with the results we have obtained, we can observe that the LDPC and polar codes have a better BER and BLER performance than the autoencoder based systems which we have implemented, and the autoencoder models need improvements if they are to be considered to be suitable alternatives to the proposed 5G implementations. When selecting physical layer implementations, processing complexity is also to be considered in order to understand advantages and disadvantages of autoencoder based systems and conventional systems.

5 CONCLUSION AND FUTURE WORK

5.1 Summary and Conclusion

Newly initiated 5G communications technology is expected to cater for the increasing communications and networking requirements of the modern society, performing significantly better than the existing 4G systems in terms of data rate, capacity, reliability, latency, energy consumption etc. Despite being very rich of expert knowledge based on statistics, information theory and mathematical modelling, existing conventional communications concepts and theories exhibit several inherent limitations in fulfilling the large data and ultra high rate communication requirements in complex scenarios. Providing fast and effective signal processing in low latency systems with reliable data transmission, having reliable data transmission in complex channels, and limited, sub-optimal performance due to the fixed block structure of the existing communications systems are some of the challenges faced. In recent years, there has been an increasing interest in applying DL concepts for the physical layer due to their flexibility, learning capability, low processing complexity and low energy consumption etc.

DL based end-to-end learning of communications systems is an emerging concept which tries to optimize both transmitter and receiver blocks in a single process in an end-to-end manner, eliminating the need for artificial block structure of the conventional communications systems. The main focus of this thesis was to investigate the performance of such DL based end-to-end communications models for single user communications scenario for the AWGN channel. Autoencoder based DL models were implemented as an equivalent to conventional communications system model and BER performances of both were compared in different system settings.

Comparing the BER performance of the uncoded communications systems with equivalent autoencoder models showed that autoencoder based systems have a comparable BER performance to the conventional uncoded BPSK or QPSK systems. Autoencoder model trained at $E_b/N_0 = 5$ dB was capable of giving comparable BER performance over the full 0-10 dB E_b/N_0 range considered, thus showing the learning capacity and capability of DL based approaches. From the simulations, BPSK equivalent autoencoder (i.e. $R = 1$ bit/channel use) was observed to have an equal or better BER performance over the 0-10 dB E_b/N_0 range while it is around 2dB better than BPSK for a BER of 10^{-5}. QPSK equivalent autoencoder (i.e. $R = 2$ bits/channel use) is better than QPSK in high E_b/N_0 range while QPSK was observed to be better in 0-5 dB E_b/N_0 range. In contrast to the conventional BPSK or QPSK communications, in the autoencoder system, transmitted signals are correlated in time, since one single message is transmitted using the allocated number of channel uses, using the optimum learnt transmission strategy. Improved BER suggests that autoencoder has learnt some joint coding and modulation utilizing the given resources which has resulted in a coding gain.

BER performance comparison of the coded communications systems with equivalent autoencoder models were then performed considering two scenarios: for BPSK communications and for higher order modulations. Same autoencoder model used for uncoded systems was used to compare the coded BPSK performance with relevant configurations. CC with both hard decision decoding and soft decision decoding using Viterbi algorithm was used as channel coding in the baseline system. BER performance was evaluated for systems with different code rates $R = \{1/2, 1/3, 1/4\}$. Autoencoder

models were observed to have comparable performance to hard decision decoding with BPSK with less than 1 dB difference. Autoencoder models were implemented with different message sizes and it was observed increasing the message size resulted in better BER performance, due to the increased degrees of freedom and flexibility for learning introduced to the model by increased message size. In contrast to the conventional system with separate coding and modulation blocks, autoencoder system learns a joint coding and modulation scheme which fits best to the channel. Here also, the learning capacity of the model is to be admired, as a single model trained at $E_b/N_0 = 5$ dB resulted in learning optimum transmission mechanisms to suit the full 0-10 dB E_b/N_0 range.

To design an equivalent system to conventional coded systems with higher order modulation schemes, a new autoencoder model was proposed which incorporated the conventional system parameters such as coding rate and modulation order etc. Simulations showed that the proposed autoencoder model is capable of achieving comparable performance to the baseline system in several instances. For $R = 1/2$ and 16-QAM scenario, equivalent autoencoder model resulted in better BER than the hard decision CC over the full 0-10 dB E_b/N_0 range while it was better than the soft decision CC in low E_b/N_0 range between 0-4 dB. For $R = 1/2$ and QPSK scenario, training the model at 5 dB did not give the expected performance. However, it was observed that training the model at 2 dB resulted in a model achieving BER performance even better than soft decision CC for low E_b/N_0 range between 0-4 dB. It shows the possibility of implementing different flexible transmission strategies based on the operating conditions (channel condition, E_b/N_0 etc.) without having a single fixed model. This flexibility of the DL based approach which can be achieved by exploiting its learning capability can be stated as an advantage over the existing conventional systems which use a fixed communication mechanism in all instances.

Processing complexity of the autoencoder based systems were also analysed in comparison to decoder complexity of the conventional systems which is considered a computationally intensive task among the other blocks. Parallel architecture of the DL models enable fast processing of information compared to the conventional systems. Also, having short block length transmission with acceptable error performance shows the potential of having DL based systems specially for low latency and low throughput applications.

Conclusively, comparable BER performance, lower processing complexity and low latency processing due to inherent parallel processing architecture, flexible structure and higher learning capacity are identified as advantages of the autoencoder based systems which show their potential and feasibility as an alternative to conventional communications systems.

5.2 Future Work

AWGN channel performance is compared under the current scope of the thesis and the autoencoder model should be extended for other fading channels to analyse its performance in fading scenarios as well. In this study we have assumed an ideal communications system with perfect timing and both carrier-phase and frequency synchronization. Generally, conventional systems are well proven to have acceptable performance in ideal system settings (with perfect timing, carrier-phase and frequency synchronization etc.), since the underlying mathematical models can be well explained in such systems compared to practical non-ideal system settings. Being able to implement DL based systems having comparable performance with respect to conventional approaches in such perfect settings show the potential of DL, whereas the real strength of DL could be exploited more in a non-ideal scenario, since DL is generally known to be well performing in situations where it is difficult to capture the underlying structures and patterns of input-output data using exact mathematical models. Thus, it is expected that DL based approaches will give better results when considering non-linear, non-ideal channel conditions and practical systems with imperfections such as timing offsets, carrier-phase and frequency synchronization issues. Therefore, further research can be carried out evaluating the performance of DL based systems in such scenarios.

Also it would be important to investigate how the autoencoder model can be extended to implement DL based systems equivalent to conventional coded systems with large block sizes and very high level modulations with comparable BLER and BER performances. In the models which we have studied, block size is not taken into consideration as the autoencoder model is formulated based on short length messages. We can find mechanisms to incorporate the block structure into the autoencoder model. We can also consider comparing the performance of the autoencoder based systems with 5G channel coding implementations like polar and LDPC codes.

6 REFERENCES

[1] Ericsson (2017) 5G systems - Enabling the transformation of industry and society. White Paper UEN 284 23-3251 rev B, Ericsson.

[2] 3GPP (2018) Study on scenarios and requirements for next generation access technologies. TR 38.913, v15.0.0, 3rd Generation Partnership Project (3GPP).

[3] Rappaport T.S. (2002) Wireless Communications: Principles and Practice. USA: Prentice-Hall; 2nd edition.

[4] Proakis J. & Salehi M. (2007) Digital Communications. McGraw-Hill Education; 5th edition.

[5] O'Shea T. & Hoydis J. (2017) An introduction to deep learning for the physical layer. IEEE Transactions on Cognitive Communications and Networking 3, pp. 563–575.

[6] O'Shea T.J., Karra K. & Clancy T.C. (2016) Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention. In: 2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), pp. 223–228.

[7] Fehske A., Gaeddert J. & Reed J.H. (2005) A new approach to signal classification using spectral correlation and neural networks. In: First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005., pp. 144–150.

[8] Nandi A. & Azzouz E. (1997) Modulation recognition using artificial neural networks. Signal Processing 56, pp. 165 – 175.

[9] Bruck J. & Blaum M. (1989) Neural networks, error-correcting codes, and polynomials over the binary n-cube. IEEE Transactions on Information Theory 35, pp. 976–987.

[10] Ortuno I., Ortuno M. & Delgado J.A. (1992) Error correcting neural networks for channels with gaussian noise. In: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, vol. 4, vol. 4, pp. 295–300 vol.4.

[11] Ibukahla M., Sombria J., Castanie F. & Bershad N.J. (1997) Neural networks for modeling nonlinear memoryless communication channels. IEEE Transactions on Communications 45, pp. 768–771.

[12] Wen C., Jin S., Wong K., Chen J. & Ting P. (2015) Channel estimation for massive mimo using gaussian-mixture bayesian learning. IEEE Transactions on Wireless Communications 14, pp. 1356–1368.

[13] Chen S., Gibson G., Cowan C. & Grant P. (1990) Adaptive equalization of finite non-linear channels using multilayer perceptrons. Signal Processing 20, pp. 107 – 119.

[14] Ibnkahla M. (2000) Applications of neural networks to digital communications – a survey. Signal Processing 80, pp. 1185 – 1215.

[15] Wang T., Wen C., Wang H., Gao F., Jiang T. & Jin S. (2017) Deep learning for wireless physical layer: Opportunities and challenges. China Communications 14, pp. 92–111.

[16] Hornik K., Stinchcombe M. & White H. (1989) Multilayer feedforward networks are universal approximators. Neural Networks 2, pp. 359 – 366.

[17] Abadi M. et al. (2015), TensorFlow: Large-scale machine learning on heterogeneous systems. URL: `https://www.tensorflow.org/`, software available from tensorflow.org.

[18] Paszke A., Gross S., Chintala S., Chanan G., Yang E., DeVito Z., Lin Z., Desmaison A., Antiga L. & Lerer A. (2017) Automatic differentiation in PyTorch. In: NIPS Autodiff Workshop.

[19] Jia Y., Shelhamer E., Donahue J., Karayev S., Long J., Girshick R.B., Guadarrama S. & Darrell T. (2014) Caffe: Convolutional architecture for fast feature embedding. In: ACM Multimedia.

[20] Zhang C., Patras P. & Haddadi H. (2019) Deep learning in mobile and wireless networking: A survey. IEEE Communications Surveys Tutorials , pp. 1–1.

[21] Goodfellow I., Bengio Y. & Courville A. (2016) Deep Learning. The MIT Press.

[22] Rumelhart D., Hinton G. & Williams R. (1989) Learning representations by back-propagating errors. Nature 323, pp. 533, 536.

[23] Chollet F. et al. (2015), Keras. `https://keras.io`.

[24] Nachmani E., Be'ery Y. & Burshtein D. (2016) Learning to decode linear codes using deep learning. In: 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 341–346.

[25] Nachmani E., Marciano E., Burshtein D. & Be'ery Y. (2017) RNN decoding of linear block codes. CoRR abs/1702.07560. URL: `http://arxiv.org/abs/1702.07560`.

[26] Gruber T., Cammerer S., Hoydis J. & t. Brink S. (2017) On deep learning-based channel decoding. In: 2017 51st Annual Conference on Information Sciences and Systems (CISS), pp. 1–6.

[27] Cammerer S., Gruber T., Hoydis J. & ten Brink S. (2017) Scaling deep learning-based decoding of polar codes via partitioning. CoRR abs/1702.06901. URL: `http://arxiv.org/abs/1702.06901`.

[28] Nachmani E., Marciano E., Lugosch L., Gross W.J., Burshtein D. & Be'ery Y. (2018) Deep learning methods for improved decoding of linear codes. IEEE Journal of Selected Topics in Signal Processing 12, pp. 119–131.

[29] Liang F., Shen C. & Wu F. (2018) An iterative BP-CNN architecture for channel decoding. IEEE Journal of Selected Topics in Signal Processing 12, pp. 144–159.

[30] Samuel N., Diskin T. & Wiesel A. (2017) Deep mimo detection. In: 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 1–5.

[31] Farsad N. & Goldsmith A.J. (2017) Detection algorithms for communication systems using deep learning. CoRR abs/1705.08044. URL: `http://arxiv.org/abs/1705.08044`.

[32] Ye H., Li G.Y. & Juang B. (2018) Power of deep learning for channel estimation and signal detection in ofdm systems. IEEE Wireless Communications Letters 7, pp. 114–117.

[33] Neumann D., Wiese T. & Utschick W. (2018) Learning the mmse channel estimator. IEEE Transactions on Signal Processing 66, pp. 2905–2917.

[34] Nandi A.K. & Azzouz E.E. (1998) Algorithms for automatic modulation recognition of communication signals. IEEE Transactions on Communications 46, pp. 431–436.

[35] Dörner S., Cammerer S., Hoydis J. & t. Brink S. (2018) Deep learning based communication over the air. IEEE Journal of Selected Topics in Signal Processing 12, pp. 132–143.

[36] O'Shea T.J., Roy T., West N. & Hilburn B.C. (2018) Physical layer communications system design over-the-air using adversarial networks. In: 2018 26th European Signal Processing Conference (EUSIPCO), pp. 529–532.

[37] MATLAB (2019) version 9.6.0 (R2019a). The MathWorks Inc., Natick, Massachusetts.

[38] Sybis M., Wesolowski K., Jayasinghe K., Venkatasubramanian V. & Vukadinovic V. (2016) Channel coding for ultra-reliable low-latency communication in 5g systems. In: 2016 IEEE 84th Vehicular Technology Conference (VTC-Fall), pp. 1–5.

[39] 3GPP (2018) Multiplexing and channel coding. TS 38.212, v15.2.0, 3rd Generation Partnership Project (3GPP).

[40] 3GPP (2018) Physical channels and modulation. TS 38.211, v15.2.0, 3rd Generation Partnership Project (3GPP).

[41] Hui D., Sandberg S., Blankenship Y., Andersson M. & Grosjean L. (2018) Channel coding in 5G New Radio: A tutorial overview and performance comparison with 4G LTE. IEEE Vehicular Technology Magazine 13, pp. 60–69.

[42] Gamage H., Rajatheva N. & Latva-aho M. (2017) Channel coding for enhanced mobile broadband communication in 5G systems. In: 2017 European Conference on Networks and Communications (EuCNC), pp. 1–6.

7 APPENDICES

Appendix 1 Deep Learning Basics

1.A Feedforward Neural Networks
1.B Convolutional Neural Networks
1.C Autoencoders
1.D Generative Adversarial Networks
1.E Network Training

APPENDIX 1 DEEP LEARNING BASICS

Appendix 1.A Feedforward Neural Networks

Deep feedforward networks, also called feedforward neural networks (feedforward NNs), or multi layer perceptrons (MLPs), are the basic type of DL models. The objective of a feedforward network is to approximate some function f^*. For example, for a classifier, $y = f^*(\mathbf{x})$ maps an input $\mathbf{x}$ to a category y. A feedforward network defines a mapping $\mathbf{y} = f(\mathbf{x}; \theta)$ and learns the value of the parameters θ that result in the best function approximation [21].

These are called **feedforward** networks since the information flows from input to the output through the intermediate computations used to define f. In feedforward NNs there are no feedback connections in which outputs of the model are fed back into itself. When feedforward NNs are extended to include feedback connections, they are called recurrent neural networks.

Feedforward NNs are given the name **networks** because they are typically made of composing many different functions together. That is, the function f which maps the input $\mathbf{x}$ to the output $\mathbf{y}$ may consists of one or more functions which connect as a chain forming the chained network structure to make the input-output mapping. For an example, there may be three functions f_1, f_2, and f_3 connected into a chain to form $f(\mathbf{x}) = f_3(f_2(f_1(\mathbf{x})))$. Here, f_1 is called the first layer, f_2 is called the second layer and so on.

The overall length of the chain gives the *depth* of the model, which has given the rise to the terminology "***deep learning***". The first layer of the feedforward network is called the **input layer**, and the final layer is called the **output layer**. Inside layers which are in-between the input and the output layers are called **hidden layers** since their behaviour is hidden to outside and the input training data does not show the desired output for these layers. Structure of a typical fully connected feedforward NN is shown in Figure Appendix 1.1.

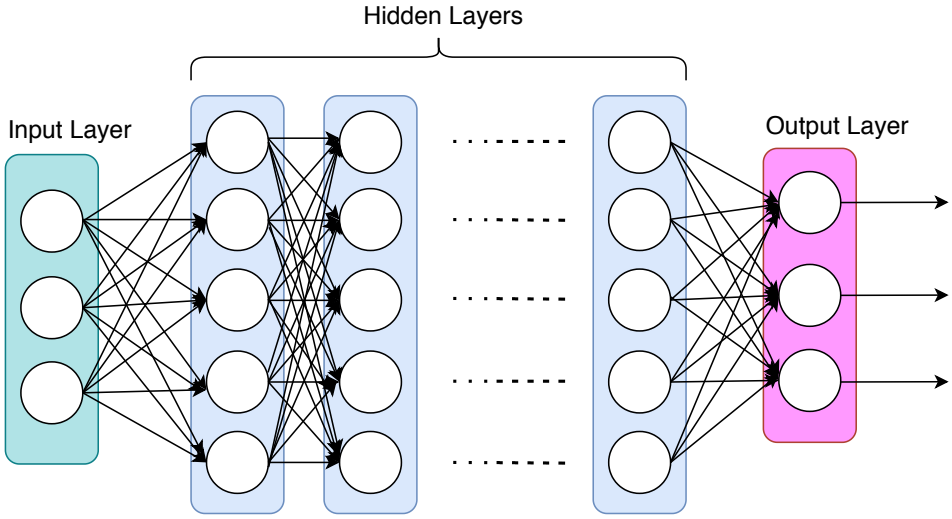

Figure Appendix 1.1. Layout of a fully connected feedforward neural network.

During the network training process, the learning algorithm decides best implementation of these **hidden layers** in order to approximate the function f^* in the optimum manner. An MLP consists of at least three layers which is the input layer, hidden layer and the output layer. Usually MLPs with more than one hidden layer are regarded as DL structures. Thus, following the above definition, a feedforward NN with L layers describes a mapping $f(\mathbf{x}_0;\theta) : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ of an input vector $\mathbf{x}_0 \in \mathbb{R}^{N_0}$ to an output vector $\mathbf{x}_l \in \mathbb{R}^{N_L}$ through L iterative processing steps as follows:

$$\mathbf{x}_l = f_l(\mathbf{x}_{l-1};\theta_l), \qquad l = 1, ..., L, \tag{2}$$

where $f_l(\mathbf{x}_{l-1};\theta_l) : \mathbb{R}^{N_{l-1}} \mapsto \mathbb{R}^{N_l}$ is the mapping performed by the l^{th} layer. This mapping depends on the output vector $\mathbf{x}_{l-1}$ from the previous layer which is fed as the input to the l^{th} layer and also the set of parameters θ_l. Furthermore, f_l can also be a function of some random variables which makes the mapping stochastic. $\theta = \{\theta_1, \theta_2, .., \theta_L\}$ denotes the set of all parameters of the network. The l^{th} layer of the network is called *dense* or *fully-connected* if $f_l(\mathbf{x}_{l-1};\theta_l)$ has the form

$$f_l(\mathbf{x}_{l-1};\theta_l) = \sigma(\mathbf{W}_l\mathbf{x}_{l-1} + \mathbf{b}_l), \tag{3}$$

where $\mathbf{W}_l \in \mathbb{R}^{N_l \times N_{l-1}}$, $\mathbf{b}_l \in \mathbb{R}^{N_l}$ and $\sigma(\cdot)$ is a called an *activation function*. The set of parameters for this l^{th} layer is $\theta_l = \{\mathbf{W}_l, \mathbf{b}_l\}$. Other than the dense layers, there are also several other types of layers which are used in feedforward NNs which cater for different requirements. Table Appendix 1.1 lists some of them along with their mapping functions and parameters [5]. The *Noise* layer has a stochastic mapping which generates a new random mapping each time it is called. That is, it adds a Gaussian noise vector with zero mean and covariance matrix $\beta I_{N_{l-1}}$ to the input, generating a different output for the same input each time it executed.

Table Appendix 1.1. Different types of layers used in neural networks

Layer Name	$f_l(\mathbf{x}_{l-1};\theta_l)$	θ_l
Dense	$\sigma(\mathbf{W}_l\mathbf{x}_{l-1} + \mathbf{b}_l)$	$\mathbf{W}_l, \mathbf{b}_l$
Noise	$\mathbf{x}_{l-1} + \mathbf{n}, \mathbf{n} \sim \mathcal{N}(0, \beta I_{N_{l-1}})$	none
Dropout	$\mathbf{d} \cdot \mathbf{x}_{l-1}, d_i \sim \mathrm{Bern}(\alpha)$	none
Normalization	e.g. $\frac{\sqrt{N_{l-1}}\mathbf{x}_{l-1}}{\|\mathbf{x}_{l-1}\|_2}$	none

The activation function $\sigma(\cdot)$ in (3) is applied individually to each element of its input vector, i.e., $[\sigma(\mathbf{u})]_i = \sigma(u_i)$. Activation function adds non-linearity to the network to make the network more powerful and adds ability to it to learn something complex and complicated form data and represent non-linear complex arbitrary functional mappings between inputs and outputs. Without this non-linearity, there would not be much of an advantage of having a stacked multiple layer structure as explained earlier. Hence, using a non-linear activation it is possible to generate non-linear mappings from inputs

to outputs. Commonly used activation functions are listed in Table Appendix 1.2 [5]. Typically in classification problems, the *SoftMax* layer is used the output layer of a network.

Table Appendix 1.2. Different types of activation functions used in neural networks

Activation Function Name	$[\sigma(\mathbf{u})]_i$	Output Range
Linear	u_i	$(-\infty, \infty)$
ReLU	$\max(0, u_i)$	$[0, \infty)$
tanh	$\tanh(u_i)$	$(-1, 1)$
sigmoid	$\frac{1}{1+e^{-u_i}}$	$(0, 1)$
softmax	$\frac{e^{u_i}}{\sum_j e^{u_j}}$	$(0, 1)$

In the model training process, the labelled data, i.e., a set of input and output vector pairs, $(\mathbf{x}_{0,i}, \mathbf{x}^{\star}_{L,i}), i = 1, ..., S$, is used to minimize the loss by adjusting the parameter set, θ. Here $\mathbf{x}^{\star}_{L,i}$ is the expected output of the NN when $\mathbf{x}_{0,i}$ is used as the input. Following equation gives the loss of the network which is tried to be minimized during the training process.

$$\mathbf{L}(\theta) = \frac{1}{S}\sum_{i=1}^{S} l(\mathbf{x}^{\star}_{L,i}, \mathbf{x}_{L,i}). \tag{4}$$

Here, $l(\mathbf{u}, \mathbf{v}) : \mathbb{R}^{N_L} \times \mathbb{R}^{N_L} \mapsto \mathbb{R}$ is the loss function and $\mathbf{x}_{L,i}$ is the actual output from the NN when $\mathbf{x}_{0,i}$ is given as the input. Commonly used loss functions include mean squared error (MSE), binary cross-entropy and categorical cross-entropy. Different loss functions are listed out in the Table Appendix 1.3. Furthermore, to train a model for a specific scenario, the loss function can be revised by adding different norms (e.g. L1, L2) of parameters or activations to the loss function. Stochastic gradient descent (SGD) algorithm is one of the most widely applied algorithms to obtain optimum sets of θ. Creating and training of deep NNs changing any of the above described parameters can be done using currently available DL libraries which are discussed in the Section 2.3.

Table Appendix 1.3. Different types of loss functions used in neural networks

Loss Function Name	$l(\mathbf{u}, \mathbf{v})$
Mean squared error (MSE)	$\|\mathbf{u} - \mathbf{v}\|_2^2$
Categorical cross-entropy	$-\sum_j u_j log(v_j)$
Binary cross-entropy	$-\frac{1}{N}\sum_j^N (u_j log(v_j) + (1-u_j) log(1-v_j))$

Appendix 1.B Convolutional Neural Networks

Convolutional neural networks (CNNs), also known as convolutional networks, are a specialized kind of neural networks to process data which has a known grid-like structure such as image data which has two-dimensional (2D) grid of pixels or time series data which has a form of one-dimensional (1D) grid. CNN is another DNN architecture developed from a fully connected feedforward network to avoid huge amounts of learnable parameters when a fully connected NN is applied for image recognition etc. A CNN consists of a set of locally connected kernals or filters to capture correlations between different data regions, instead of having full connections between layers. As the name implies, a CNN employs a mathematical operation called convolution which is a special kind of linear operation. Thus, CNNs are simply NNs which use convolution instead of general matrix multiplication in at least a single layer of the network. The convolution operation can be denoted by the following equation for each location $\mathbf{p}_y$ of the output $\mathbf{y}$ [20],

$$\mathbf{y}(\mathbf{p}_y) = \sum_{\mathbf{p}_G \in \mathbb{G}} \mathbf{w}(\mathbf{p}_G) \cdot \mathbf{x}(\mathbf{p}_y + \mathbf{p}_G), \tag{5}$$

where $\mathbf{p}_G$ denotes all positions in the receptive field $\mathbb{G}$ of the convolutional filter $\mathbf{w}$ and the weights $\mathbf{w}$ are shared across different locations of the input map. Convolution leverages three main ideas which improves a traditional DL system: sparse interactions, parameter sharing and equivarient representations which enables CNNs to be more effective. And also they provide ways of working with variable sized inputs [21].

A typical CNN consists of single or multiple convolution layers, pooling layers and then one or two fully connected layers at the end of the network. In the pooling layer, the neurons in the feature maps are grouped together to obtain a single representation of each feature map such as computing the mean value (average pooling) or maximum value (max pooling) of each feature map. Thus pooling layers enable reducing the number of parameters substantially before using the fully connected network. A basic structure of a CNN is shown in the Figure Appendix 1.2. The network is trained end-to-end with labelled training data similar to training of a normal feedforward NN.

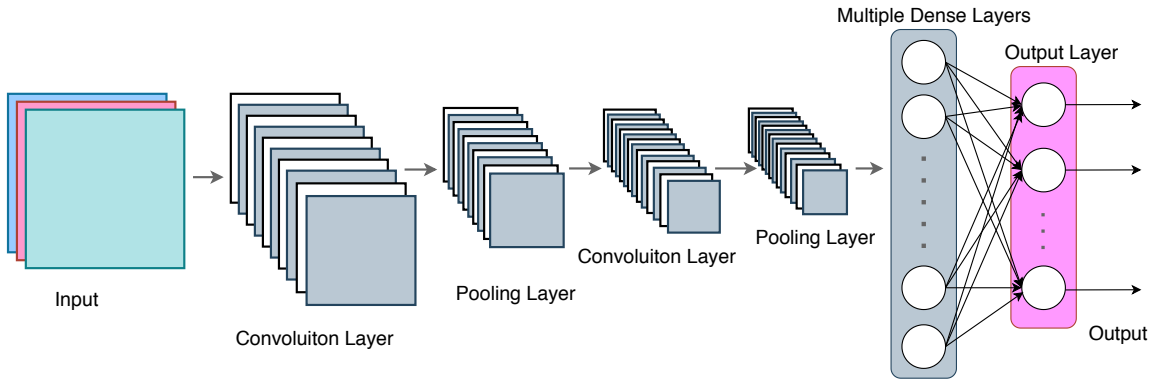

Figure Appendix 1.2. An example layout of a convolutional neural network (CNN).

Appendix 1.C Autoencoders

An autoencoder is a type of artificial neural network (ANN) which tries to reconstruct its input at the output in an unsupervised manner. Thus, it falls into the category of unsupervised machine learning algorithms. Internally, it has a hidden layer $\mathbf{h}$ that describes a code used to represent the input $\mathbf{x}$. Since the network is trained to reconstruct its inputs, the hidden layer is forced to try to learn good representations of the inputs. The network can be viewed as consisting of two parts: an encoder function $\mathbf{h} = f(\mathbf{x})$ and a decoder that produces a reconstruction $\hat{\mathbf{x}} = g(\mathbf{h})$.

The learning process of the autoencoder can be described as minimizing the reconstruction loss as follows:

$$\mathbf{L}(\mathbf{x}, g(f(\mathbf{x}))), \tag{6}$$

where $\mathbf{L}$ is a loss function such as MSE which penalizes $g(f(\mathbf{x}))$ for being different to the input $\mathbf{x}$.

Traditionally, autoencoders were used for dimensionality reduction or feature learning. Autoencoders are closely related to principal component analysis (PCA) since PCA also tries to reduce dimensionality of input data in an unsupervised manner by minimizing the reconstruction error. However, autoencoders can represent both linear and non-linear transformation in encoding and decoding, thus has more flexibility, while PCA generally performs linear transformation. Autoencoders can be layered to form DL network due to its network representation. There can be multiple hidden layers where a deep NN is formed. Layout of an autoencoder NN is shown in the Figure Appendix 1.3. Autoencoders thus can be thought as a special case of feedforward networks, and may be trained with all of the same techniques, typically mini-batch gradient descent following gradients computed by back-propagation.

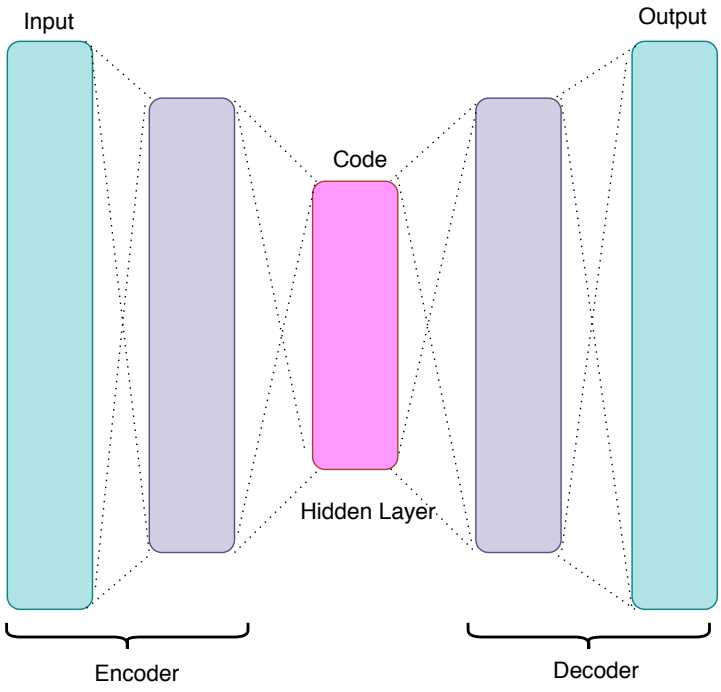

Figure Appendix 1.3. Layout of an autoencoder.

Depending on the application the autoencoder is used, there are different types of autoencoders such as undercomplete autoencoders, sparse autoencoders, denoising autoencoders, contractive autoencoders etc.

Appendix 1.D Generative Adversarial Networks

Generative adversarial network (GAN) is a type of generative modelling approach based on differentiable generator networks, which can produce data that follows certain target distribution, p_{data}. Generative adversarial networks are based on a game theoretic scenario where the generator network competes against its adversary, and thus a GAN consist of a generator $g(\cdot)$ and a discriminator $d(\cdot)$. Structure of a GAN is shown in the Figure Appendix 1.4. The discriminator attempts to differentiate between real data and fake data generated by the generator, while the generator tries to generate credible data to mislead the discriminator into making mistakes.

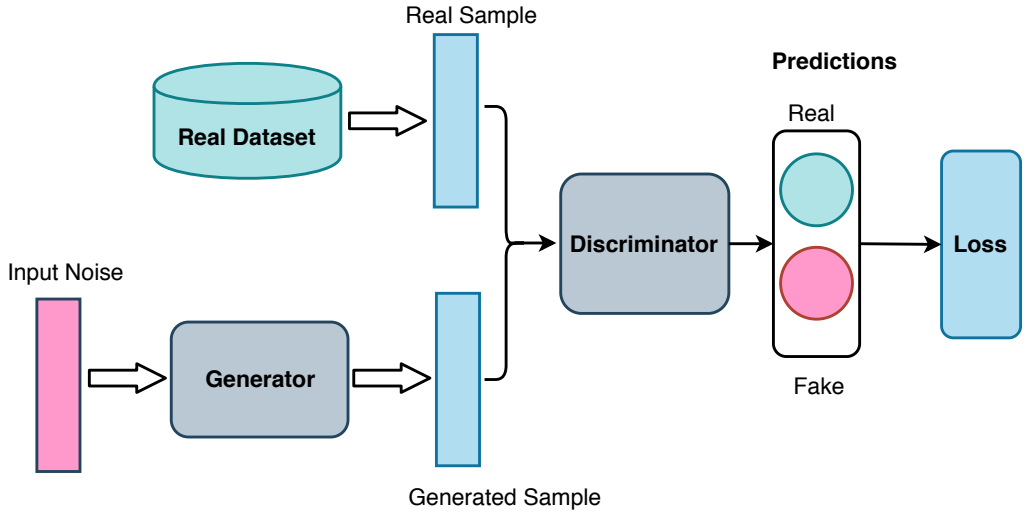

Figure Appendix 1.4. Structure of a generative adversarial network (GAN).

The generator directly produces samples $\mathbf{x} = g(\mathbf{z}; \theta^{(g)})$ and the discriminator outputs a probability value given by $d(\mathbf{x}; \theta^{(d)})$, which indicates the probability that $\mathbf{x}$ is a real training sample instead of a fake sample generated by the generator model. Therefore, a min-max two players game is introduced between the generator g and the discriminator d, and the min-max optimization objective can be given as

$$\arg\min_{g}\max_{d} \nu(\theta^{(g)}, \theta^{(d)}) = \mathbb{E}_{\mathbf{x}\sim p_{data}} \log d(\mathbf{x}) + \mathbb{E}_{\mathbf{x}\sim p_{model}} \log(1 - d(\mathbf{x})). \tag{7}$$

A Conditional GAN extends the above formulation by providing some extra conditioning information $\mathbf{m}$, where the conditioning information is fed to both the generator g and the discriminator d as an additional input. Then the min-max optimization objective changes to the form

$$\arg\min_{g}\max_{d} \nu(\theta^{(g)}, \theta^{(d)}) = \mathbb{E}_{\mathbf{x}\sim p_{data}} \log d(\mathbf{x}|\mathbf{m}) + \mathbb{E}_{\mathbf{x}\sim p_{model}} \log(1 - d(\mathbf{x}|\mathbf{m})). \tag{8}$$

Appendix 1.E Network Training

Training of NNs involves finding the optimum parameters for each of the layers in the network which minimizes a desired loss function such as the loss function given in 4 for a simple feedforward NN. The differentiable architecture of DNNs allows learning the optimum model parameters which minimizes the loss function using gradient descent (GD) methods through back-propagation, following the fundamental chain rule [22]. Having a large number of hidden layers and neurons result in having several other parameters to be determined and thereby makes the network implementation difficult. Vanishing gradients, slow convergence of the network, getting stuck in a local minimum are some of the problems which are faced during network training process [15]. Vanishing gradient problem where the gradients of the loss function approaches zero, which makes the network training difficult is solved by introducing new activation functions such as rectified linear units (ReLU) [15].

A modified version of the classic GD algorithm is used to achieve faster convergence and to reduce the computation complexity, which is known as the stochastic gradient descent algorithm which is widely used in network training. SGD starts with a random initialization of the parameters $\theta = \theta_0$ and then updates them iteratively as

$$\theta_{t+1} = \theta_t - \eta \nabla \tilde{\mathbf{L}}(\theta_t), \tag{9}$$

where $\eta > 0$ is the learning rate and $\tilde{\mathbf{L}}(\theta_t)$ is an approximation of the loss function computed for a randomly selected mini-batch of training samples $\mathcal{S}_t \subset \{1, 2, .., S\}$ of size S_t at each iteration, given as

$$\tilde{\mathbf{L}}(\theta_t) = \frac{1}{S_t} \sum_{i \in \mathcal{S}_t} l(\mathbf{x}^{\star}_{L,i}, \mathbf{x}_{L,i}). \tag{10}$$

It is noted that gradient complexity can be significantly reduced by selecting S_t small compared to S, while still reducing the weight update variance [5]. In order to avoid converging the model to local optimal solutions and to further increase training speed, different adaptive learning rate algorithms such as Adagrad, RMSProp, Momentum, and Adam have been proposed [15].

Another challenge of network training is that even though the network gets trained well and performs well for the training data, it may give poor performance for testing data due to overfitting for the training data. To avoid overfitting, different approaches like early stopping, regularization and dropout schemes have been proposed which results in getting acceptable performance in both training and testing data [15].