Channel estimation and equalization using generative adversarial networks

AJ Oosthuizen

orcid.org/0000-0001-7471-6384

Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor: Prof ASJ Helberg

Co-supervisor: Prof MH Davel

Graduation: June 2023

Student number: 28413113

Declaration

I, Andrew John Oosthuizen, hereby declare that the dissertation entitled "Channel estimation and equalisation using generative adversarial networks" is my own original work and has not already been submitted to any other university or institution for examination.

A.J. Oosthuizen

Student number: 28413113

Signed on the 25th day of November 2022 at Potchefstroom.

Acknowledgements

This research was performed within the MUST research group of the North-West University, which is a member of the Centre for Artificial Intelligence Research (CAIR), as well as the Telenet research group of the North-West University, which is a member of the Telkom Centre of Excellence (COE). It was supervised by Professor Albert Helberg and Professor Marelie Davel in a joint venture between these two research groups.
I want to extend gratitude to:

- My wife Lucinda, whom I married at the time this study took place and who without I would not have had the courage to pursue this study in the passionate manner I had.
- My supervisors, who provided me with the opportunity and freedom to pursue topics I found of interest. I am also grateful for the many hours spent reviewing my work to ensure that only the best quality work was released.
- Ulrike Janke, who always ensured that all administrative tasks were dealt with swiftly and keeps the heart of the MUST group beating.

To my family, thank you for all the encouraging words and calming phone calls in times of great stress; without you, this dissertation would not be half the quality it is today. Finally, I would like to acknowledge NWU Hockey, where I spent most of my time as an undergraduate and some time as a postgraduate student; and where I met my loving wife. The lessons learned and character built on the NWU Astro turf in Potchefstroom prepared me for the undertaking of this dissertation and my life to follow.

Abstract

Channel State Information (CSI) estimators are used in everyday communication systems to estimate channel impairments that affect transmitted data, and to equalise the impaired data to a more accurate state. However, this can be a complex task as channels cause many non-linear impairments to transmitted data.

In this study, we construct a simulated environment to investigate the effects of channel impairments on CSI data in Long Term Evolution (LTE) environments. Using the simulated environment, we also generate datasets with which we investigate the ability of several deep learning architectures to estimate CSI. We extend this investigation to adversarial training techniques that have had success on computer vision tasks that are similar to CSI estimation. These trained deep learning networks are evaluated in several wireless communication environments to investigate the effect of adversarial training on network performance. We start this analysis by investigating networks in the Single-In Single-Out (SISO) environment before moving to Multi-Antenna (MA) environments. In this process, we find that the performance of adversarially trained networks in an MA environment deviates from the expected performance indicated in the SISO training environment. Finally, we show that adversarial training has the potential to train better performing CSI estimators without increasing the computational complexity of the network when implemented in a wireless communications system.

Keywords: *Channel state information, Deep learning, Adversarial training, Multiple-In Multiple-Out, Long-term evolution*

Contents

List of Figures **ix**

List of Tables **xiv**

List of Abbreviations **xvii**

1 Introduction **1**

1.1 Background 1

1.2 Problem statement 3

1.3 Research objectives 3

1.4 Research questions 4

1.5 Project scope 4

1.6 Research methodology 5

1.7 Dissertation overview 6

1.8 Publications and timeline 7

2 Background **9**

2.1 Introduction 9

2.2 Wireless systems 10

2.2.1 Basic wireless communication principles 10

2.2.2 Wireless channel impairments 13

2.2.3 Wireless communication standards 16
2.2.4 Wireless CSI estimation 20
2.2.5 MA environments 24
2.3 Deep learning 28
2.3.1 Neural architectures 28
2.3.2 Training of neural networks 30
2.3.3 Adversarial neural networks 33
2.4 Applications of deep learning for CSI estimation in literature 35
2.4.1 Reasons for using deep learning 35
2.4.2 Related studies 36
2.5 Adversarial architectures used in this study 37
2.5.1 SRGAN 37
2.5.2 Adversarial ResNet 39
2.6 Conclusion 40

3 Wireless system simulation **41**

3.1 Introduction 41
3.2 Physical layer simulations 42
3.3 Resource grid construction 43
3.4 Channel impairments 45
3.4.1 Simple channel 46
3.4.2 Delay profiles 47
3.4.3 Doppler spread 48
3.4.4 Antenna noise and interference 49
3.5 Channel estimation using LS and LTE-MMSE 50
3.6 Multi-antenna environments 52

3.7 Simulation results 54
3.8 Conclusion 58

4 CSI estimation using MLPs **59**

4.1 Introduction 59
4.2 Architecture 60
4.3 Dataset 60
4.4 Training protocol 62
4.5 Results 63
4.5.1 Feature representation 63
4.5.2 Effect of hyperparameter choices 66
4.5.3 Observed performance 70
4.6 Conclusion 71

5 CSI estimation using ResNet **72**

5.1 Introduction 72
5.2 ResNet architecture 73
5.3 Data samples 73
5.4 Noise generalisation ability 74
5.4.1 Training protocol 75
5.4.2 Generalisation results 75
5.4.3 Generalisation comparison 76
5.5 Combined SNR dataset 77
5.5.1 Experimental protocol 77
5.5.2 Distributed dataset results 78
5.5.3 PICDB 78

5.5.4 Final results 79
5.6 Conclusion 80

6 CSI estimation using GANs **82**
6.1 Introduction 82
6.2 Dataset 83
6.2.1 Enhanced even noise distributed dataset 83
6.2.2 Dataset evaluation 83
6.3 Training protocol 85
6.4 Architectures 86
6.4.1 Adversarial ResNet 86
6.4.2 SRGAN 87
6.4.3 Modified SRGAN 88
6.5 Adversarial training on small input size 89
6.6 Adversarial training over increased input sizes 90
6.6.1 ResNet using different input sample sizes 91
6.6.2 Adversarial training using different input sample sizes 92
6.7 Conclusion 95

7 Analysis of multi antenna environments **96**
7.1 Introduction 96
7.2 Simulation setup 97
7.3 Results 98
7.3.1 Receiver diversity results 98
7.3.2 Transmitter diversity results 102
7.4 Discussion 104

7.4.1 SISO vs MA . . . 105
7.4.2 Receiver diversity discussion . . . 105
7.4.3 Enhanced noise problem in receiver diversity . . . 107
7.5 Benchmark comparison . . . 109
7.6 Conclusion . . . 111

8 Conclusion **113**
8.1 Introduction . . . 113
8.2 Key findings . . . 114
8.3 Observations . . . 115
8.4 Contributions . . . 116
8.5 Future work . . . 117
8.6 Conclusion . . . 118
References . . . 119

A Supplementary content **126**
A.1 Appendix: Chapter 3 . . . 126
A.2 Appendix: Chapter 5 . . . 128
A.3 Confidence . . . 129
A.3.1 BER . . . 129
A.3.2 Models . . . 129
A.4 Published work . . . 132
A.4.1 Exploring CNN-Based Automatic Modulation Classification Using Small Modulation Set . . . 132
A.4.2 Multi-Layer Perceptron for Channel State Information Estimation: Design Considerations . . . 138
A.4.3 Adversarial Training for Channel State Information Estimation in LTE Multi-Antenna Systems . . . 145

List of Figures

2.1 Constellation diagrams of 4- and 16-Quadrature Amplitude Modulation (QAM) modulation schemes depicting the number represented by carrier symbols with certain Quadrature and In-phase (QI) alterations. 12

2.2 A depiction of how subcarriers are orthogonally overlapped in the frequency domain to create the time domain Orthogonal Frequency division multiplexing (OFDM) signal. 13

2.3 A depiction of how Least Squares (LS) CSI estimation is calculated within a resource grid using pilot symbols with green blocks representing CSI estimates. 22

2.4 Depicted is a representation of a Multiple-Input Multiple-Output (MIMO) system with the transmitter antennas on the left and the receiver antennas on the right with independent channels connecting the antennas. 25

2.5 The training structure of a typical Generative Adversarial Network (GAN) model 34

2.6 Illustration of the modified Super Resolution Generative Adversarial Network (SRGAN) network structure applied by Zhao *et al.* [16]. 40

3.1 Physical layer blocks implemented to create the wireless communication simulation 43

3.2 A representation of a subframe with pilot positions shown in green 44

3.3 Figures depicting the amplitude and phase shifts of CSI in the frequency domain for a simple channel with no impairments. 46

3.4 Power spectral density plots depicting the effect delay profiles have on transmitted OFDM symbols in the frequency domain. 47

3.5 Figures depicting the amplitude shifts of CSI in the frequency domain for multipath channels with different delay profiles. 48

3.6 Figures depicting the amplitude shifts of CSI in the time domain for a multipath channel Extended Vehicular A (EVA) delay profile under different Doppler spreads. 48

3.7 4-QAM modulated QI points after an OFDM symbol has been subjected to various Signal-to-Noise-Ratios (SNRs) of Additive White Gaussian Noise (AWGN). The colours signify the bits represented by the specific QI symbol. 49

3.8 QI plots of CSI obtained using different estimation methods. Each colour represents an individual OFDM symbol of the plotted subframe. 51

3.9 Demodulated 4-QAM symbols after being equalised with estimated CSI. The colours indicate the bits represented by the specific QI point. 52

3.10 Bit Error Rate (BER) of a receiver diversity system with Extended Pedestrian A (EPA) delay profile depicting the average SISO channel performance of the system along with the performance of two and four receiver antennas. 53

3.11 BER of a transmitter diversity system with EPA delay profile depicting the average SISO channel performance of the system along with the performance of one and two receiver antennas. 54

3.12 The BER of a SISO system with an EVA delay profile and 100 Hertz (Hz) Doppler spread. We depict the performance of 4 and 16-QAM modulations using LS, Minimum Means Squared Error (MMSE) and perfect CSI estimations. 55

3.13 The BER of a four antenna receiver diversity system with an EVA delay profile and 100 Hz Doppler spread. We depict the performance of 4 and 16-QAM modulations using LS, MMSE and perfect CSI estimations. . . . 56

3.14 The BER of a two receiver antenna transmitter diversity system with an EVA delay profile and 100 Hz Doppler spread. We depict the performance of 4 and 16-QAM modulations using LS, MMSE and perfect CSI estimations. 56

3.15 Here we depict the noise enhancement problem. Subfigure a shows two CSI conditions in red and blue, with green being their original position. Subfigure b shows the result of equalisation on AWGN when perfect CSI is applied. 58

4.1 The process of creating features as input for the Multilayer Perceptron (MLP) can be seen on the left of this figure, while CSI reconstruction from the MLP's output is illustrated on the right. 61

4.2 CSI estimate produced by the MLP in attenuated channel conditions and the resulting equalised symbols obtained when equalising with the estimated CSI. 71

5.1 Illustration of the modified SRGAN's generator network developed by Zhao *et al.* [16] and used in this study. 73

5.2 An illustration of a CSI constellation diagram with 10 Decibel (dB) antenna noise being translated to an input data sample with three features. Each QI symbol translates to a 1x3 point in the data sample and the data sample dimensions correspond to the subframe size selected, in this case, a slot. . . 74

5.3 Percentage Increase of Correctly Decoded Bits (PICDB) of the CSI generated by estimators trained on the Even Noise Distribution Dataset (ENDD), High Noise Distribution Dataset (HNDD) and Low Noise Distribution Dataset (LNDD) with respect to single SNR trained CSI estimators, shown over the test set. 79

5.4 BER of the combined dataset training method, single SNR trained networks, MMSE and LS estimators over the SNR range, shown over the test set. 80

6.1 PICDB of the data demodulated by CSI generated from networks trained on SNR distributed datasets with respect to Enhanced Even Noise Distributed Dataset (EENDD) CSI estimators, shown over the test set with 16-QAM symbols using 12x7 input sample sizes. 84

6.2 A representation of the composition of the content and adversarial loss in Adversarial ResNet. 87

6.3 A representation of the composition of the content and adversarial loss in SRGAN. 88

6.4 A representation of the composition of the content and adversarial loss in Modified Super Resolution Generative Adversarial Network (MSRGAN). . 89

6.5 PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 12x7, shown over the test set. 90

6.6 PICDB of the CSI generated by non-adversarially trained ResNet with respect to an LS estimation for three input sample sizes, shown over the test set. 91

6.7 Figures depicting the CSI estimates of four 12x7 estimators, one 24x14 estimator and the perfect CSI for the same channel conditions. 92

6.8 PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 24x14, shown over the test set. 93

6.9 PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 120x14, shown over the test set. 94

7.1 The interconnection of environments when importing trained models from Python to Matlab. 98

7.2 PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 12x7 in a receiver diversity environment. 99

7.3 PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 24x14 in a receiver diversity environment. 100

7.4 PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 120x14 in a receiver diversity environment. 100

7.5 PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 12x7 in a transmitter diversity environment. 102

7.6 PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 24x14 in a transmitter diversity environment. 103

7.7 PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 120x14 in a transmitter diversity environment. 103

7.8 The number of CSI estimations made by networks that are smaller than 5e-2 for both Quadrature (Q) and In-phase (I) components over the entire SNR range, using the test set. 106

7.9 The number of symbols larger than 3 for both Q and I components equalised using different networks' CSI estimations over the SNR range, using the test set. 107

7.10 This figure indicates the effect small CSI estimations and enhanced noise can have in a receiver diversity system, by comparing symbols equalised using slightly larger and smaller CSI estimates. 108

7.11 BER of 16-QAM equalised symbols in a SISO environment using ResNet, MMSE and LS CSI estimators over the SNR range, shown over the test set. 109

7.12 BER of 16-QAM equalised symbols in a receiver diversity environment with four receiver antennas using ResNet, MMSE and LS CSI estimators over the SNR range. 110

7.13 BER of 16-QAM equalised symbols in a transmitter diversity environment with two receiver antennas using ResNet, MMSE and LS CSI estimators over the SNR range. 111

A.1 A BER comparison between the results of Mehmood et al. [70] for a SISO channel and the simulation created for this study. The error bars represent a 95% confidence interval. 127

A.2 PICDB, with standard deviations, of the CSI generated by adversarially trained networks compared to non-adversarially trained ResNet for an input sample size of 24x14 in a receiver diversity environment with 2 receiving antennas. 130

A.3 PICDB, with standard deviations, of the CSI generated by adversarially trained networks compared to non-adversarially trained ResNet for an input sample size of 24x14 in a receiver diversity environment with 4 receiving antennas. 131

List of Tables

2.1 An example delay profile with a tap being a specific path followed by the signal, excess tap delay the time the signal arrives after the reference time (being the first tap), and relative power the strength of the arriving signal on the specific tap. 14

2.2 Wireless system parameters of an LTE transmission as indicated by the 3rd Generation Partnership Project (3GPP) standard. 17

2.3 Terminology used in resource grids 17

2.4 Terminology used in resource grids of a 10 Mega-Hertz (MHz) LTE transmission 18

2.5 The EPA delay spread is shown in this table with a tap being a specific path followed by the signal, excess tap delay the time the signal arrives after the reference time (being the first tap), and relative power the strength of the arriving signal on the specific tap. 18

2.6 The EVA delay spread is shown in this table with a tap being a specific path followed by the signal, excess tap delay the time the signal arrives after the reference time (being the first tap), and relative power the strength of the arriving signal on the specific tap. 19

2.7 The Extended Typical Urban (ETU) delay spread is shown in this table with a tap being a specific path followed by the signal, excess tap delay the time the signal arrives after the reference time (being the first tap), and relative power the strength of the arriving signal on the specific tap. 20

3.1 Parameters used in LTE physical layer simulation for modulation and resource block construction. 45

4.1 Channel conditions used in the simulation for each dataset generated in this chapter. 62

4.2 The value ranges which the initial hyperparameter grid search uses to search for optimal architectural and training hyperparameters. 63

4.3 Predicted CSI angle of two OFDM signals compared to the target CSI angle over the same OFDM signals. 64

4.4 Best performing structural hyperparameters validation set Mean Squared Error (MSE) loss for MLP trained on noiseless data. 67

4.5 Best performing structural hyperparameters validation set MSE loss results for MLP trained on noiseless data and no batch normalisation layers. . . . 68

4.6 Validation set MSE loss results over grouped training parameter for noisy data with QI values as features. 69

4.7 The same setup as in Table 4.6 but including batch normalisation. 69

4.8 BER of MLP networks trained on one delay profile train set and applied to the same or other delay profile test sets. 70

4.9 BER of different equalisation methods on the Doppler dataset with various amounts of Doppler spread. 70

5.1 The value ranges that the initial hyperparameter grid search uses to search for optimal training hyperparameters. 75

5.2 BER of ResNet networks trained on different SNR and tested over all datasets' test sets. 76

5.3 BER of MLP networks trained on different SNR levels and tested over all datasets' test sets. 76

5.4 The number of SNR samples included in each distributed dataset. 77

5.5 BER of MLP networks trained on different SNRs and tested over all datasets' test sets. 78

6.1 Average PICDB over all SNRs of ResNet networks trained on different noise distributed datasets compared to the EENDD. 84

6.2 Hyperparameter ranges used for the initial sweeps before being expanded to wider sweeps. 85

6.3 Average PICDB over all SNRs of adversarially trained networks trained on different input sample sizes compared to a ResNet of the same input sample size. 94

7.1 Average PICDB over all SNRs of adversarially trained networks in a receiver diversity environment with two receiving antennas compared to a ResNet network. 101

7.2 Average PICDB over all SNRs of adversarially trained networks in a receiver diversity environment with four receiving antennas compared to a ResNet network. 101

7.3 Average PICDB over all SNRs of adversarially trained networks in a transmitter diversity environment with one receiving antenna compared to a ResNet network. 104

7.4 Average PICDB over all SNRs of adversarially trained networks in a transmitter diversity environment with two receiving antennas compared to a ResNet network. 104

A.1 Hyperparameter ranges used for the grid search for an input sample size of 12x7. 128

A.2 Hyperparameter ranges used for the grid search for an input sample size of 24x14. 128

A.3 Hyperparameter ranges used for the grid search for an input sample size of 120x14. 128

List of Abbreviations

3GPP 3rd Generation Partnership Project

AWGN Additive White Gaussian Noise

BCE Binary Cross-Entropy

BER Bit Error Rate

CGAN Conditional Generative Adversarial Network

CNN Convolutional Neural Network

CSI Channel State Information

dB Decibel

DFT Discrete Fourier Transform

DNN Deep Neural Network

EGC Equal Gain Combing

EENDD Enhanced Even Noise Distributed Dataset

ENDD Even Noise Distribution Dataset

EPA Extended Pedestrian A

ETU Extended Typical Urban

EVA Extended Vehicular A

FFT Fast Fourier Transform

FLOPS Floating Point Operations Per Second

GAN Generative Adversarial Network

HNDD High Noise Distribution Dataset

Hz Hertz

I In-phase

IFFT Inverse Fast Fourier Transform

ITU International Telecommunication Union

KHz Kilo-Hertz

LNDD Low Noise Distribution Dataset

LMMSE Linear Minimum Means Squared Error

LR Learning Rate

LS Least Squares

LTE Long Term Evolution

MA Multi-Antenna

MHz Mega-Hertz

MIMO Multiple-Input Multiple-Output

MLP Multilayer Perceptron

MMSE Minimum Means Squared Error

MRC Maximal Ratio Combining

MSE Mean Squared Error

MSRGAN Modified Super Resolution Generative Adversarial Network

NLP Natural Language Processing

NN Neural Network

OFDM Orthogonal Frequency division multiplexing

ONNX Open Neural Network Exchange

OSI Open Systems Interconnection

OSTBC Orthogonal Space-Time Block Code

PICDB Percentage Increase of Correctly Decoded Bits

PReLU Parametric Rectified Linear Unit

Q Quadrature

QAM Quadrature Amplitude Modulation

QI Quadrature and In-phase

ReLU Rectified Linear Unit

RMSProp Root Mean Squared Propagation

SACAIR Southern African Conference for Artificial Intelligence Research

SATNAC Southern Africa Telecommunication Networks and Applications Conference

SGD Stochastic Gradient Descent

SISO Single-In Single-Out

SNR Signal-to-Noise-Ratio

SR Super Resolution

SRGAN Super Resolution Generative Adversarial Network

VGG Visual Geometry Group

Chapter 1

Introduction

In this chapter we introduce the basis of this study and outline how the rest of this study is structured.

1.1 Background

As the world becomes more connected, communication protocols have developed to provide more capacity and bandwidth to our wireless communication systems. However, these protocols and wireless communication methods suffer from channel impairments that reduce the effective amount of data that can be transmitted. Channel State Information (CSI) estimators attempt to estimate the CSI, over which signals are transmitted to combat these channel impairments. CSI estimation is a complex task that has received much attention [1], [2] in the telecommunications field.

Some CSI estimators estimate channel conditions using prior CSI knowledge [1]. The CSI estimations obtained from these estimators allow the receiving device to equalise data that has been impaired due to channel conditions. This equalisation process plays a

prominent role in modern single and Multi-Antenna (MA) systems to ensure that received data corresponds as much as possible to what was transmitted.

Many approaches [1], [2] have been taken to develop CSI estimators capable of adequately estimating CSI while reducing the resources needed. CSI estimators need to keep computational resources such as the number of floating point operations required and the memory size of algorithms in mind while also keeping the overhead of unnecessary data transmitted as low as possible. For these reasons, linear estimators are frequently used in practice, even though they are not the best performing method in terms of CSI estimation, but they do keep computational resources and overheads low. More complex statistical methods, such as Minimum Means Squared Error (MMSE), have also been proposed that increase the CSI estimation performance at the cost of computational resources. These methods can be used if the necessary equipment is available for their deployment.

In recent years deep learning models such as Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) have also been developed for CSI estimation and have achieved CSI estimation performance competing with the MMSE estimation method. Deep learning models typically upscale an initial CSI estimation from prior data available on the channel conditions. However, these models are computationally expensive and need to sacrifice estimation performance if they are to be adapted for mobile device usage.

With deep learning gaining traction in many fields, specifically computer vision [3], many new techniques have emerged that are applicable to other fields. One of these methods is adversarial training [4], which, in the case of this study, introduces a secondary network to the training process that aids the training of the initial network. These adversarial models are referred to as Generative Adversarial Networks (GANs). This training method has shown the ability to increase network performance for a specific task without increasing model sizes. For example, adversarial training has shown promising results in image generation, denoising and upscaling tasks [5], [6].

Adversarial training has primarily been applied in the telecommunications field to mimic channels [7], [8]. However, the upscaling ability of adversarial networks for CSI estimation

has only been touched on by the research community and could have the potential to further increase network performance without needing additional computational resources or data overhead.

1.2 Problem statement

Channel estimation and equalisation are used within a telecommunications system to improve the number of successfully transmitted signals. In conjunction with MA technology, this method combats channel impairments that hinder the effective transmission of signals by estimating these impairments and equalising the received signals accordingly. Deep learning methods [9]–[13] have been applied to the field, showing promising results. GANs, however, have yet to be thoroughly researched for CSI estimation while being implemented in many other fields of study [6], [14], where they have achieved state-of-the-art results. In previous work [15], [16], GANs are used to augment existing CSI estimation techniques in Single-In Single-Out (SISO) channels. We hypothesise that using adversarial training in CSI estimation can improve the performance of existing neural network architectures. To this end, we propose investigating the implementation of adversarial training techniques for CSI estimation in MA environments.

1.3 Research objectives

To address this problem statement, we set the following objectives in this study:

- Investigate channel impairments found in wireless communication environments by simulating channels with a variety of impairments to generate CSI datasets.
- Make use of different data features and dataset compositions to investigate how deep learning networks trained on different CSI datasets compare with one another.

- Apply adversarially and non-adversarially trained networks to the CSI estimation task.
- Investigate adversarial network implementations in MA environments for the CSI estimation task.

1.4 Research questions

To address the research objectives we ask the following research questions:

- How should we construct datasets to investigate the behaviour of Neural Networks (NNs) for the CSI estimation task?
- Does adversarial training improve on traditional training for the CSI estimation task and if so under which conditions?
- How do single-channel NN CSI estimators perform under antenna diversity configurations?

1.5 Project scope

Given the broad fields of machine learning, deep learning and telecommunications, and the methods used in these fields, it is essential to limit the scope of the study. Therefore, to ensure that the research stays focused and that all research questions are answered concisely, we will restrict ourselves to the following scope:

- Physical layer communication: As CSI estimation is implemented in the physical layer of telecommunications systems [17], we investigate and simulate only physical layer communication in this study.

- MA systems: Communication systems can communicate using many different techniques when multiple antennas are available. In this study, we focus only on MA systems that use antenna diversity techniques as they focus on increasing throughput through CSI diversity.

- Wireless standard: Many wireless standards dictate how signals are transmitted in the physical layer and how CSI estimation is deployed. For this reason, we use the Long Term Evolution (LTE) [18] standard that is well documented and has been in public use for many years, making it relevant to the problem.

- NN architectures: Due to the vastness of the CSI estimation task and data availability, many deep learning model architectures could be applied to the task. We chose only to include networks with fully connected and convolutional layers in this study based on results found in previous work and the applicability of adversarial training to these architectures.

1.6 Research methodology

This dissertation is an empirical analysis of adversarial training methods. The main aspects of this study consist of:

- **Literature study**: A review of necessary telecommunication and CSI estimation concepts and methods is done. The principles needed for the understanding and training of NNs are also reviewed. Finally, the application of deep learning to CSI estimation and architectures of NNs of importance to this study is discussed.

- **Selecting an appropriate development environment**: In this study, we use the PyTorch Python package to train and evaluate NNs. Matlab® is used to develop a LTE wireless communication simulation using the LTE[1] and deep learning[2] toolbox. These two environments are used in parallel to create channel-specific datasets for

[1] `https://www.mathworks.com/products/lte.html`
[2] `https://www.mathworks.com/products/deep-learning.html`

training and to test networks in a simulated communication system environment. In addition, Weights and Biases[3] is used for experiment tracking and NN training scheduling as many networks are trained in parallel on different computing resources.

- **Experimental procedure**: To effectively evaluate NNs' ability to estimate CSI we:
  - Conduct a hyperparameter sweep to optimise selected hyperparameters and train several networks with different initialisation seeds.
  - Evaluate network performance over a range of Signal-to-Noise-Ratio (SNR) for specific CSI conditions using simulated data.
  - Compare networks with other networks and methods on identical datasets and channel conditions using suitable metrics.
- **Evaluation**: To evaluate adversarial training for CSI estimation we compare the results of adversarial trained networks to non-adversarial trained networks of identical architectures in SISO and MA environments.

1.7 Dissertation overview

The dissertation has the following structure:

- Chapter 2 discusses concepts and methods used within this study, as well as related work. We also discuss applications of deep learning on CSI estimation.
- Chapter 3 develops the simulation that is used to investigate channel impairments and create datasets for NNs to be trained and evaluated.
- Using these datasets, Chapter 4 investigates different structural and training hyperparameters for CSI estimation. This chapter uses an MLP model, which is one of the simplest deep learning architectures, to help further understand CSI data.

[3] `https://wandb.ai/`

- Chapter 5 introduces NN architectures with convolutional layers that are capable of extracting spatial-temporal information from inputs.
- Chapter 6 compares adversarial training to traditional training in a SISO environment, using the dataset and NN architecture investigated in Chapter 5.
- Chapter 7 evaluates adversarial training and traditional training methods in MA environments, using the networks trained in the SISO environment in Chapter 6.
- Chapter 8 concludes the dissertation with a summary of the findings.

1.8 Publications and timeline

The research described in this study led to three publications, which can be found in Appendix A.4:

- "Exploring CNN-Based Automatic Modulation Classification Using Small Modulation Sets" [19], an exploratory study that led to this work, was published at the Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2021 conference.
- "Multi-Layer Perceptron for Channel State Information Estimation: Design Considerations" [20] was published at the SATNAC 2022 conference.
- "Adversarial Training for Channel State Information Estimation in LTE Multi-Antenna Systems" [21] was accepted for publication at Southern African Conference for Artificial Intelligence Research (SACAIR) 2022, and will be published in Communications in Computer and Information Science, volume 1734.

This study took place according to the following timeline:

- Student registration took place in March 2021.

- All research described in this study occurred between March 2021 and November 2022.
- A colloquium took place in August 2021 leading to the registration of the title of this study in the same month.
- The first conference paper [19] was published in November 2021[4].
- Ethics screening was performed by the faculty scientific committee as part of the title registration, and a "No risk" rating was recommended. The ethics clearance was finalised by the faculty research ethics committee on 11 April 2022.
- The second conference paper [20] was published in August 2022.
- The third paper [21] was accepted for publication in a special journal edition of the SACAIR outputs, to be published in December 2022.

Some of the previously published work is included in this dissertation: The 2022 SATNAC paper is adapted and used for Chapter 4 of this study, where we apply MLP architectures to the CSI estimation task. Chapters 5, 6 and 7 extend the work to be published in the 2022 SACAIR publication, specifically the work done on CNN training, input sample size selection and adversarial training.

[4]This paper received the SATNAC 2021 best paper award.

Chapter 2

Background

This chapter provides background on concepts and methods used within this study. We also discuss how these methods interact by investigating applications of deep learning on CSI estimation.

2.1 Introduction

In this chapter, we provide background relevant to this study. First, we discuss concepts of importance to CSI estimation within wireless communication systems, such as basic wireless communication principles, wireless channel impairments, communication standards, commonly used CSI estimation methods, and MA environments. This is followed by a discussion on Deep Neural Networks (DNNs), how they are trained and how adversarial architectures are implemented. Finally, we discuss the use of DNNs for the CSI estimation task and review directly related studies. Note that sections of this chapter has been published in Oosthuizen *et al.* [19], [20].

2.2 Wireless systems

Wireless communication systems have become part of our daily lives through cell phones, computers and wearable technology. These systems connect the world as we know it by enabling mobile and consistent connection to the internet. However, as data needs increase [22], wireless communication technology must evolve to ensure this need is met. In this study, we aim to contribute to CSI estimation, a core technology in wireless communication systems.

This subsection discusses the basic concepts of wireless communication systems, wireless channels, communication standards, CSI estimation and MA environments. These concepts are discussed as they are used throughout the rest of the study.

2.2.1 Basic wireless communication principles

Wireless communication systems consist of many methods to ensure that our data gets transmitted securely and in a timely manner. Nevertheless, the fundamental goal of wireless communication systems remains to transmit data from one location to another over a wireless channel. This simple task requires many technologies and techniques to be used to create the high-speed communications systems we use daily. In this subsection, we discuss the fundamental principles of wireless communication, and the role modulation plays in this process.

Fundamentals

The first and most important concept of wireless communication is how we transmit and receive data between two antennas. From a physics point of view, wireless communication is able to take place because electromagnetic waves are produced when electrons move through a conductor [23]. When this flow of electrons is controlled, we can generate meaningful waves that travel over a wireless medium such as air. At some point in this

wireless medium, another antenna exists that produces an electron flow, similar to the initial electron flow in a conductor when exposed to electromagnetic waves.

We, however, need to refine this communications process to ensure that as much data can be transmitted as possible. To do this, we need to utilise the frequency spectrum's finite resource maximally. Spectrum is a finite resource since when two antennas create electromagnetic waves in a similar space and time without regard for one another, interference takes place over the medium, rendering both waves irrecoverable. To combat this interference, waves are generated at different frequencies, allowing us to recover data by inspecting a specific frequency on the medium. We call these waves 'carrier waves' [24].

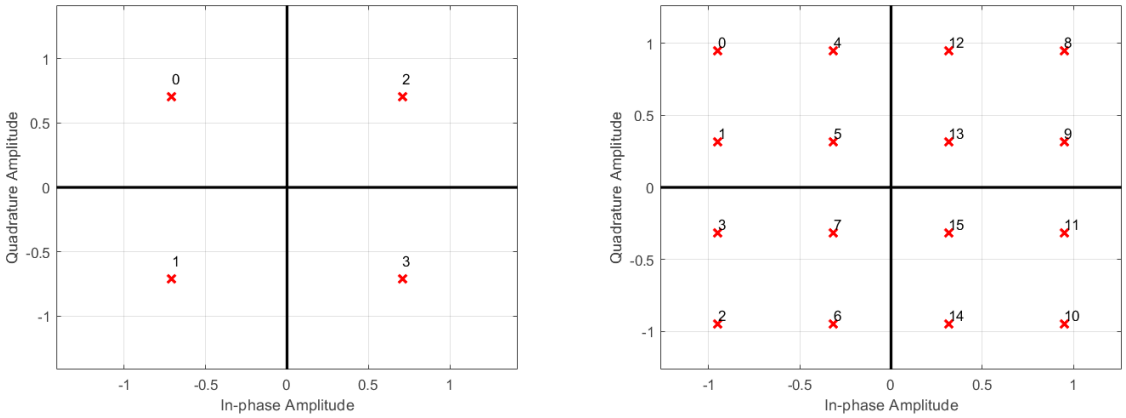
Data is embedded in carrier waves by altering the carrier wave, usually a sine wave of a specific frequency. We can thus extract data from the carrier wave at the receiving antenna by comparing the received wave to the expected carrier. Carrier waves are altered by changing their amplitude and phase when transmitted. By incorporating both the amplitude and phase, or Quadrature and In-phase (QI), components of a signal, we can represent binary bits using altered sine waves. The simplest example is amplitude shift keying modulation, where a binary one is represented by the carrier wave having an unchanged amplitude. A binary zero is represented by a carrier with a much smaller or no amplitude [24].

Modulation schemes

As the accuracy and ability of hardware advanced, we were able to more reliably alter and detect alterations to carrier waves, enabling us to use more complex modulation schemes. Quadrature Amplitude Modulation (QAM) [24] enables us to represent several bits with a single carrier signal.

Constellation diagrams represent how the carrier signal is altered from the original carrier. When demodulating QI symbols we demodulated received symbols to the bits of the closest QAM demodulation point. Bit errors thus occur when received QI symbols move to demodulation regions other than their transmitted regions.

QAM modulation is applied for several orders, with higher orders representing more bits per QI symbol. Examining Figure 2.1 we observe that 16-QAM is populated more densely than 4-QAM but can transmit more bits at any given time. Lower-order modulations are used in channels with many impairments as demodulation points are spaced further apart, thus making classification more robust if the QI components of the carrier are altered when transmitted.



(a) 4-QAM (b) 16-QAM

Figure 2.1: Constellation diagrams of 4- and 16-QAM modulation schemes depicting the number represented by carrier symbols with certain QI alterations.

While QAM modulation provides a more robust and frequency-efficient way of transmitting data than amplitude modulation, it is still not effective enough to provide us with the spectrum usage needs of modern-day communication systems [22], [24]. For this reason, we multiplex the already modulated symbols to Orthogonal Frequency division multiplexing (OFDM) symbols.

OFDM is a digital multicarrier modulation scheme often used to send multiple data streams over the same carrier [25], [26]. OFDM allows multiple carriers, called subcarriers, to carry signals from the transmitter to the receiver. Subcarriers are created by allocating a small portion of the allowed broadcasting spectrum to a subcarrier. OFDM is made more efficient by overlapping these subcarriers orthogonally in the frequency domain so that signals do not interfere with each other, as shown in Figure 2.2. After the OFDM signal has been created in the frequency domain, an Inverse Fast Fourier Transform (IFFT) is applied

to allow the symbol to be transmitted in the time domain. Upon receiving the OFDM symbol, the receiver applies Fast Fourier Transform (FFT) to separate the subcarriers and retrieve individually modulated signals that can be further demodulated [27].

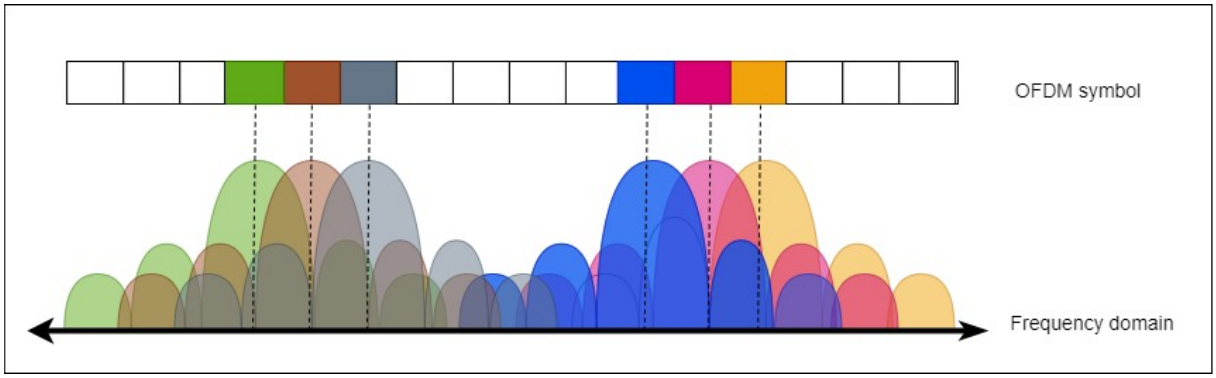


Figure 2.2: A depiction of how subcarriers are orthogonally overlapped in the frequency domain to create the time domain OFDM signal.

2.2.2 Wireless channel impairments

Unlike wired channels, wireless channels are unguided connections and entail several complex environmental factors. Environmental factors play a prominent role in wireless communication because signals are transmitted through free space as waves [23], [24].

These environmental factors introduce fading to transmissions. Fading can be placed into two categories: large-scale fading and small-scale fading [27]. We focus on small-scale fading [28] as large-scale fading is categorised by path loss and shadowing, commonly found in signals that travel large distances. In this subsection, we discuss frequency fading, time fading and Additive White Gaussian Noise (AWGN) as they are commonly observed impairments in small-scale fading models.

Frequency fading

Multipath effects commonly cause frequency fading [27] and occur because signals act as waves and simultaneously take many routes from the transmitting antenna to the receiving

antenna. The received signal is a summation of all the different paths a signal takes from the transmitter to the receiver. As the path lengths and effects of the paths differ, the summation of the signals will often experience phase changes and varying signal strength relative to the transmitted signal. Signals with a line-of-sight are often less affected by this phenomenon as the direct path delivers a strong and clear signal that is only slightly disrupted by the other weaker paths the signal follows.

We categorise multipath fading as either frequency-selective or flat. If a channel is frequency selective, the amount of fading will vary over the channel bandwidth, while flat fading experiences a constant fading over the channel bandwidth.

Multipath fading is often characterised using power delay profiles. Delay profiles characterise channels by providing several different fading paths with different relative delays and average power for each path. An example of a delay profile can be found in Table 2.1.

Table 2.1: An example delay profile with a tap being a specific path followed by the signal, excess tap delay the time the signal arrives after the reference time (being the first tap), and relative power the strength of the arriving signal on the specific tap.

Tap	Excess tap delay (ns)	Relative power (Decibel (dB))
1	0	0.0
2	30	-1.0
3	60	-2.0
4	90	-3.0

OFDM is often used for wireless communication systems to guard against multipath effects. Since fading and interference are often frequency-specific, only some subcarriers will be affected instead of the whole carrier, leading to higher successful data transmission rates. In addition, by adding guard intervals, OFDM further guards against multipath effects and inter-symbol interference [29].

Time fading

Signals can also experience fading in the time domain, similar to what they do in the frequency domain. Variations of signal quality in the time domain are closely correlated to Doppler spread, caused by relative movement between the transmitter and the receiver [27].

We classify time selective fading as slow or fast fading based on the Doppler spread's aggressiveness and the transmitted symbol's symbol time. If the channel impulse, which is correlated to the Doppler spread, changes fast compared to the baseband transmission signal, fast fading is experienced. When fast fading is experienced, each transmitted symbol will experience different channel conditions from the previous symbol in the time domain.

Additive white Gaussian noise

Noise is an ever-present impairment found in most wireless and wired communications. AWGN appears in received signals due to noise created by electronic circuits and other noise-inducing factors in the environment [24]. Heat-induced electronic noise, general vibrations and weak signals from other transmitted signals in the frequency band are often the main reason for AWGN found on received signals.

AWGN is a channel impairment assumed to have a constant power spectral density over the given channel's bandwidth and a normal distribution with a mean of zero. We measure noise using the SNR metric that describes the ratio between noise and signal power in the frequency domain.

Wireless transmission noise can be described mathematically by obtaining the SNR as a ratio from the SNR expressed in dB as SNR in Equation 2.1. Using this SNR, along with the number of transmit antennas n_{Tx} and the size of the FFT used for OFDM modulation $Nfft$, we normalise random numbers $RandN$ obtained from a normal distribution with

a mean of zero in Equation 2.2[1].

$$SNR = 10^{\frac{SNR_{dB}}{20}} \tag{2.1}$$

$$\mathbf{AWGN} = \frac{\mathbf{RandN}}{\sqrt{2n_{Tx}Nfft} * SNR} \tag{2.2}$$

2.2.3 Wireless communication standards

Wireless communication has obtained widespread acceptance and has been integrated to many tasks. However, these tasks frequently have different requirements from wireless communications, such as preferring reliability over speed. For this reason, wireless communication standards, such as WiFi [30], LORaWAN [31] and LTE [18], were written that describe how wireless communication takes place over a specific frequency band. In addition, each of these standards describe a different set of transmission and data handling practices that best serve its intended task.

Long-term evolution

Due to the many available standards, we select a single standard on which to base our work. Therefore, we select the 3rd Generation Partnership Project (3GPP) LTE standard [18], a widely used standard with well-defined transmission rules to use as a reference in this study.

LTE is a fourth-generation wireless standard that transmits data in the radio wave spectrum. LTE is known for its long-range capabilities and is used to connect mobile devices to the internet via base stations. The LTE standard depicts several modes of operation and uses OFDM modulation for downlink communication from the base station to the user device.

[1] `https://www.mathworks.com/help/lte/index.html?s_tid=CRUX_topnav`

A summary of the LTE specifications [18] of importance can be found in Table 2.2.

Table 2.2: Wireless system parameters of an LTE transmission as indicated by the 3GPP standard.

Parameter	LTE specification
Carrier frequencies	400 MHz to 5000 MHz
Antennas	4x2, 2x2, 1x2, 1x1
Modulation	4-QAM to 256-QAM
Bandwidth (MHz)	1.4, 3, 5, 10, 15, 20
Subcarrier spacing	15 Kilo-Hertz (KHz)

LTE resource grids

The LTE standard transmits OFDM symbols over a specified channel bandwidth, limiting the number of carriers placed in the frequency domain as each subcarrier takes up a space of 15 KHz. In Table 2.3, it is shown how the number of subcarriers linearly scales to the channel bandwidth. Table 2.3 also depicts the utilised frequency bandwidth, which describes how much bandwidth is utilised by subcarriers. We note that the bandwidth utilisation is 90%, leaving 10% bandwidth for guard bands which are used so as not to interfere with neighbouring transmissions.

Table 2.3: Terminology used in resource grids

Channel bandwidth (Mega-Hertz (MHz))	1.4	3	5	10	15	20
Number of resource blocks	6	15	25	50	75	100
Number of subcarriers	72	180	300	600	900	1200
Used bandwidth (MHz)	1.08	2.7	4.5	9	13.5	18

Within this bandwidth of subcarriers and OFDM symbols, certain naming conventions have been employed to avoid confusion. Considering a channel bandwidth of 10 MHz, the naming conventions used in Table 2.4 refer to specific data blocks:

Table 2.4: Terminology used in resource grids of a 10 MHz LTE transmission

Allocation name	Size (subcarrier × OFDM symbol)
Subcarrier symbol	1x1
OFDM symbol	600x1
Slot	12x7
Resource block or subframe	12x14
Resource grid	600x14

LTE delay profiles

The 3GPP standard for LTE has specified three delay profiles provided by the International Telecommunication Union (ITU) [32] for multipath simulations within the LTE standard. Each delay profile specifies a different environment of operation and increases channel complexity from Extended Pedestrian A (EPA) to Extended Typical Urban (ETU).

The EPA delay profile used for LTE is a modified version of the EPA delay profile specified by the ITU, shown in Table 2.5. The adapted EPA delay profile for LTE simulations contains 7 taps with each tap having an increase in relative power attenuation and a maximum path delay of 410 ns. This delay profile is used to characterise slow-moving user equipment, up to 3 km/hour, such as pedestrians walking, and for this reason, is usually simulated with a maximum Doppler spread of up to 5 Hertz (Hz).

Table 2.5: The EPA delay spread is shown in this table with a tap being a specific path followed by the signal, excess tap delay the time the signal arrives after the reference time (being the first tap), and relative power the strength of the arriving signal on the specific tap.

Tap	Excess tap delay (ns)	Relative power (dB)
1	0	0.0
2	30	-1.0
3	70	-2.0
4	90	-3.0
5	110	-8.0
6	190	-17.2
7	410	-20.8

The Extended Vehicular A (EVA) delay profile used for LTE is an extended version of the EPA delay profile specified by the ITU, shown in Table 2.6. This extended delay profile contains 9 taps with each tap having a varied response in relative power attenuation and a maximum path delay of 2 510 ns. This delay profile is used to characterise a moving user equipment, around 120 km/hr, such as fast-moving vehicles, and for this reason is usually simulated with a maximum Doppler spread of up to 70 Hz.

Table 2.6: The EVA delay spread is shown in this table with a tap being a specific path followed by the signal, excess tap delay the time the signal arrives after the reference time (being the first tap), and relative power the strength of the arriving signal on the specific tap.

Tap	Excess tap delay (ns)	Relative power (dB)
1	0	0.0
2	30	-1.5
3	150	-1.4
4	310	-3.6
5	370	-0.6
6	710	-9.1
7	1 090	-7.0
8	1 730	-12.0
9	2 510	-16.9

The ETU delay profile used in LTE is based on the Global System for Mobile communication Typical Urban Model, shown in Table 2.7. This adapted delay profile contains 9 taps with each tap having a varied response in relative power attenuation and a maximum path delay of 5 000 ns. This delay profile is used to characterise moving user equipment, around 350 km/hr, such as very fast-moving vehicles and for this reason, is usually simulated with a maximum Doppler spread of up to 300 Hz.

Table 2.7: The ETU delay spread is shown in this table with a tap being a specific path followed by the signal, excess tap delay the time the signal arrives after the reference time (being the first tap), and relative power the strength of the arriving signal on the specific tap.

Tap	Excess tap delay (ns)	Relative power (dB)
1	0	-1.0
2	50	-1.0
3	120	-1.0
4	200	0.0
5	230	0.0
6	500	0.0
7	1 600	-3.0
8	2 300	-5.0
9	5 000	-7.0

2.2.4 Wireless CSI estimation

With channel impairments degrading the quality of received signals, methods such as channel estimation and equalisation are employed to increase the amount of correctly demodulated data. In this subsection, we discuss the implementation of channel estimation as a whole, followed by discussions of several CSI estimation methods and how we measure CSI estimators' performance.

Perfect CSI

CSI estimation is the practice of predicting the collective impairments over a given channel, while channel equalisation uses the CSI to equalise distorted signals to a more decodable and less impaired state [27]. CSI consists of a QI symbol that represents the change experienced by the receiving signal due to channel conditions. Understanding what impairments are found in the channel enables us to recover each subcarrier better and reduce the number of bit errors or Bit Error Rate (BER) when demodulating symbols.

We refer to CSI over a single time step as h, and CSI over multiple time steps as $\mathbf{h}$. In

Equation 2.3 we can see that perfect CSI $h_{Perfect}$ can be obtained by dividing the received signal y by the transmitted signal x. By dividing the received signal by the CSI, we can obtain the estimated transmitted signal.

$$h_{Perfect} = \frac{y}{x} \tag{2.3}$$

When implementing several channels we represent the CSI over a single time step as $\mathbf{H}$ with channel conditions between specific antenna pairs as $h_{nTx,nRx}$ with nTx representing a specific transmit antenna and nRx a specific receive antenna.

$$\mathbf{H} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \tag{2.4}$$

A commonly used method for obtaining the CSI is to place pilot symbols within the transmitted signals. Pilot symbols are transmitted signals known both to the transmitter and the receiver and can be used by the receiver to estimate CSI through methods like Least Squares (LS) and MMSE [33].

Least squares

A simple way to use pilot symbols to estimate CSI is to interpolate between the known CSI symbols provided by the pilot symbols. LS calculates the CSI estimation, over several time and frequency steps H, in OFDM symbols by interpolating between the subcarriers with known CSI. As seen in Figure 2.3 this bilinear interpolation is first done within the OFDM symbols containing pilot symbols, generating CSI in the frequency domain and then interpolated over time. This provides us with a complete LS CSI estimate.

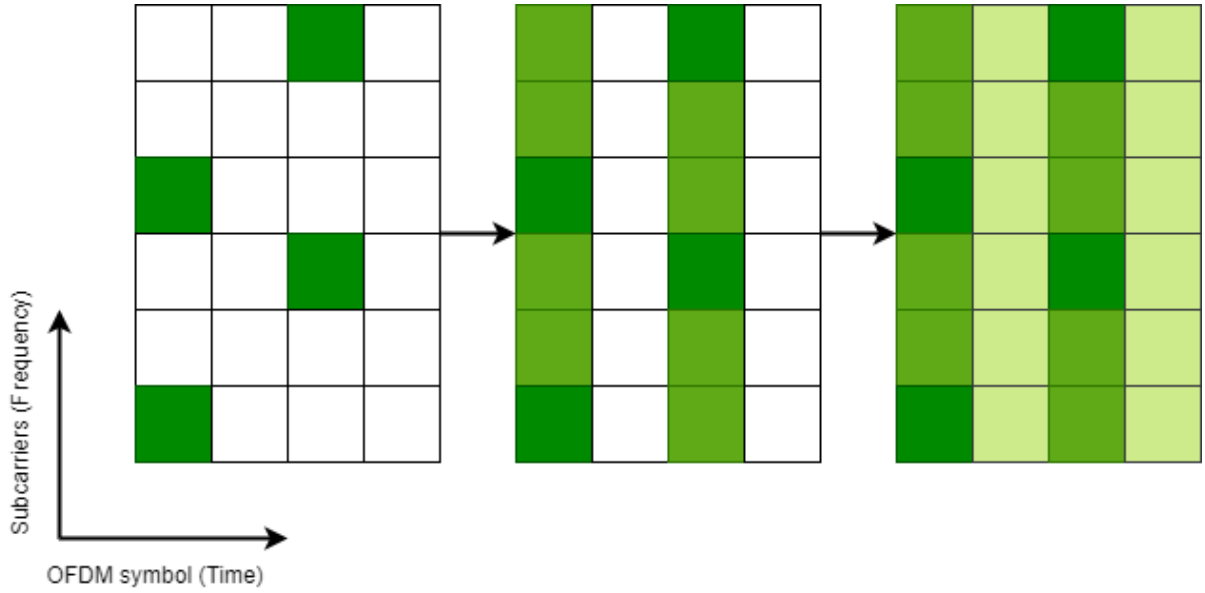


Figure 2.3: A depiction of how LS CSI estimation is calculated within a resource grid using pilot symbols with green blocks representing CSI estimates.

LS CSI estimation provides a basic CSI estimation at a minimal computational cost but may, at times, not capture non-linear impairments due to the linear nature of interpolation. In addition, pilot symbols also suffer from AWGN, making the starting points of the interpolation inaccurate.

LTE-MMSE

Standards have adapted standard LS CSI estimations into estimation methods that better fit the standard's wireless environment and available computational resources. For example, LTE has a downlink CSI estimator and equaliser that is used in place of LS. This method has slightly more Floating Point Operations Per Second (FLOPS) but provides advantages such as denoising.

LTE downlink estimation is done by obtaining pilot points from the received data and then averaging these pilots using time and frequency windows[2] [18]. Downlink estimation removes noise from the CSI generated by pilot points leading to a more accurate interpo-

[2] `https://www.mathworks.com/help/lte/ug/channel-estimation.html`

lation. In addition, this estimation method further creates virtual pilots based on existing pilots to increase the information available during interpolation.

After interpolation has been completed and a CSI estimate $\hat{H}$, over time and frequency subcarriers, has been obtained the data is equalised using a modified MMSE equaliser that applies the inverse of a noise estimation N to the data:

$$\hat{X} = \frac{Y}{\hat{H}} N^{-1}. \tag{2.5}$$

where Y is the received symbols and $\hat{X}$ the estimated symbols over a set time and frequency. We refer to this method as LTE-MMSE moving forward.

Other methods

Several methods exist that improve the LS estimation by using known channel statistics. However, obtaining these channel statistics is difficult in unknown channels and implementing them over the LS estimation increases the computational resources needed.

MMSE makes use of a weight matrix W to improve on channel estimate $\hat{H}$ by finding an improved CSI estimate $\tilde{H}$, making use of the autocorrelation R matrix of $\hat{H}$ and the cross-correlation matrix between the true channel and estimated channel in the frequency domain [27], [33], as seen in Equation 2.6:

$$\tilde{H} = W\hat{H} = R_{H\hat{H}} R_{\hat{H}\hat{H}}^{-1} \hat{H}. \tag{2.6}$$

This weight matrix W requires the time and frequency characteristics of the channel to be known.

Discrete Fourier Transform (DFT)-based channel estimation uses LS estimation to eliminate noise outside the maximum channel delay. Implementing a discrete Fourier transform improved channel estimation and can eliminate noise from CSI estimations. This method,

however, requires the maximum channel delay to be known and requires more FLOPSs than other methods due to the use of Fourier transforms [27].

CSI estimation metrics

In this study, we mainly use three metrics to evaluate our CSI estimators' performance. We use BER and Percentage Increase of Correctly Decoded Bits (PICDB), discussed in Section 5.5.3, to indicate the effect of CSI estimation on demodulated bits, while Mean Squared Error (MSE) is a direct measurement of our estimated CSI.

The most frequently used metric is BER which indicates the number of bit errors in decoding a set number of received bits. BER can thus be used as an indicator of CSI estimation accuracy, as the more accurate the CSI being used for equalisation, the fewer bit errors are made in demodulation. We calculate BER as shown in Equation 2.7:

$$\text{BER} = 1 - \frac{\text{Correctly classified bits}}{\text{Total bits}} \tag{2.7}$$

Lastly, we use MSE to compare the features of our CSI estimation to the real CSI's features. MSE is mainly used for neural network optimisation as system performance is based on the amount of correctly demodulated bits and not on how accurate CSI estimations are. However, MSE can be used as a comparative metric between CSI estimators as we expect a correlation between how accurate CSI estimators' features are and BER.

2.2.5 MA environments

Modern-day wireless implementations need more capacity between devices to keep up with demand [22]. In order to obtain these higher throughput rates, multiple antennas are used on both transmitters and receivers, as can be seen in Figure 2.4, to increase the available bandwidth available to devices [27]. By using this setup, instead of a single transmitter and single receiving antenna, we spatially diversify the transmitted signal

and linearly increase the available bandwidth per antenna by creating multiple new paths between the transmitter and receiver.

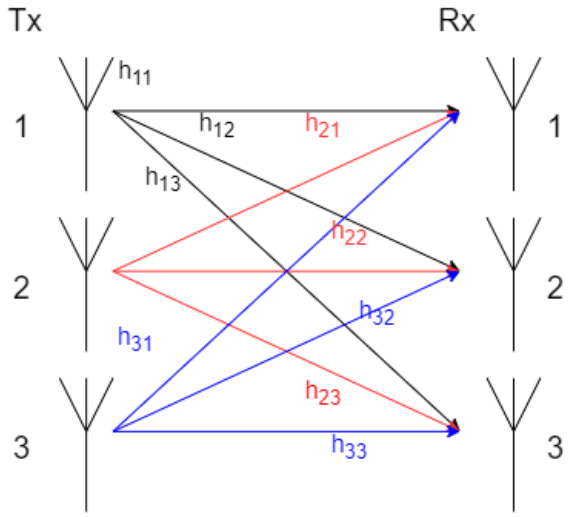


Figure 2.4: Depicted is a representation of a Multiple-Input Multiple-Output (MIMO) system with the transmitter antennas on the left and the receiver antennas on the right with independent channels connecting the antennas.

The new paths created by the multiple antennas help alleviate multipath impairments by providing multiple paths, with some statistically having fewer impairments than the rest, and providing more capacity to the network by increasing the quality of received signals and thus the amount of correctly decoded data. Antenna diversity can be implemented in several ways such as receiver diversity or transmitter diversity, with each of these methods working from different principles and obtaining different results. In this subsection, we discuss the implementation of receiver and transmitter diversity.

Receiver diversity

Receiver diversity environments consist of communication between a transmitter with a single antenna and a receiver with multiple antennas [27]. The transmitting antenna thus transmits a single data stream to multiple antennas over multiple channels. These channels each have different channel conditions, meaning that some channels deliver poorer

transmissions to the receiving antenna than others. We can leverage this diversity using three different methods.

The first method, called selection combination [27], selects the best channel, measured by an SNR estimator, and only uses that channel's received data and discards the inferior channel data. Secondly, we can average over all of the antennas' equalised data to generate a single data stream; this is called Equal Gain Combing (EGC) [27]. An EGC implementation using two receive antennas is depicted in Equation 2.8.

$$X = \frac{\frac{Y_1}{H_1} + \frac{Y_2}{H_2}}{2} \tag{2.8}$$

Lastly, we can make use of a weight vector w containing measured SNR for each channel. The weight vector is used to assign weights to equalised data in the combining process, this method is called Maximal Ratio Combining (MRC) [27]. As can be seen in Equation 2.9 equalised data with lower SNRs gets assigned lower weights and contributes less to the final equalised signal y_{MRC}.

$$y_{MRC} = \sum_{i=1}^{N_R} w_i^{MRC} y_i \tag{2.9}$$

From these three methods, MRC performs the best. EGC, being a special case of MRC with all weights being equal, is the only method that does not require an SNR estimator and will be used in receiver diversity applications in this study.

Transmitter diversity

Transmitter diversity is implemented using multiple transmitter antennas but can also consist of either single or multiple receive antennas. Once again, multiple channels form, each with their own independent CSI.

We make use of Orthogonal Space-Time Block Code (OSTBC) in this study to implement transmitter diversity. Specifically, we make use of an Alamouti encoder [27], [34] that encodes a single data stream to two separate data streams to be transmitted over two

antennas. These two data streams are encoded to be orthogonal to one another using the mapping in Equation 2.10.

$$X = \begin{bmatrix} x_1 & -x_2^* \\ x_2 & x_1^* \end{bmatrix} \tag{2.10}$$

To explain the equalisation process we make use of a system with two transmit antennas and two receive antennas. Having two transmit antennas means that the two transmissions interfere at the receiving antenna, making pilot extraction difficult. To alleviate this problem, pilot symbols are placed in different subcarriers for different transmit antennas and the corresponding subcarrier in the other antenna is left unoccupied. Using the pilot symbols we are now able to obtain the CSI for each antenna pair.

Using this CSI between antenna pairs we are able to recombine the orthogonally transmitted data by combining two neighbouring time instances. We first define time step one's equation as Equation 2.11:

$$\begin{bmatrix} y_{1,1} \\ y_{1,2} \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \tag{2.11}$$

and time step two's equation as Equation 2.12:

$$\begin{bmatrix} y_{2,1} \\ y_{2,2} \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} -x_2^* \\ x_1^* \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}. \tag{2.12}$$

Combining these two time steps' information at the receiver's side and implementing the complex conjugate on the second time step's noise, CSI and received symbols we obtain Equation 2.13:

$$\begin{bmatrix} y_{1,1} \\ y_{1,2} \\ y_{2,1}^* \\ y_{2,2}^* \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \\ h_{1,1}^* & -h_{1,2}^* \\ h_{2,1}^* & -h_{2,2}^* \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_1^* \\ n_2^* \end{bmatrix}. \tag{2.13}$$

After discarding noise from the equation as we accept that it cannot be estimated we obtain the equalised symbols from solving the matrix multiplication in Equation 2.14, with $\mathbf{H}^H$ being the hermitian transpose of the combined channel matrix over two time

steps.

$$\begin{bmatrix} \hat{x_1} \\ x_2^* \end{bmatrix} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H \begin{bmatrix} y_{11} \\ y_{12} \\ y_{21}^* \\ y_{22}^* \end{bmatrix} \tag{2.14}$$

A similar process is followed for a system with two transmitting antennas and a single receiving antenna.

2.3 Deep learning

Deep learning is a machine learning method in which DNNs are used to extract features from data for classification or regression purposes [35]. Deep learning models have been found to excel in computer vision, Natural Language Processing (NLP), telecommunication, image generation and many other fields [36]–[38]. The term deep learning was derived from deep neural networks which are networks consisting of many connected layers. In this section, we discuss neural architectures, the training protocol of neural networks and adversarial neural networks as they are of importance to this study.

2.3.1 Neural architectures

Neural networks consist of several layers that make up their architectures and give specific characteristics and abilities to the entirety of the architecture.

Fully-connected layers

MLP architectures typically consist of nodes. Nodes are elements with multiple inputs, which are calculated from previous node values multiplied by trainable weights [39], [40]. A node's value is a summation of all inputs from connected nodes in the MLP architecture which has been passed through a non-linear activation function. Nodes are placed next

to one another to create layers of nodes that are all connected to nodes in previous layers but are connected by different trainable weights. MLPs can have many layers, making them deeper networks. These layers can also contain many nodes in a single layer creating wider layers.

Convolution layers

CNN architectures use convolutional layers that pass a windowed filter, with trainable weights for each cell, over a multidimensional input or previous layer [35], [41]. The ability of the convolution layer to generate a value from several localised values in the previous layer enables CNNs to perceive spatial correlations in multidimensional data in a way that MLPs cannot. However, due to how filters need to move over dataspaces and the number of filters typically employed in CNNs they need notably more computational resources to train and to deploy.

Pooling functions [39], [41] are also employed in CNN architectures to provide summary statistics of regions of convolutional outputs. An example of pooling is maximum pooling where only the largest number found within a windowed filter is propagated to the output. Pooling functions are used to reduce dimensionality within networks and for certain feature extraction purposes.

Residual blocks [35], [42] have also been implemented in conjunction with convolution layers. Residual blocks consist of several convolution layers being applied to an input, with the input then being added back to the output of the sequence of convolution layers. This implementation was introduced to combat vanishing gradients in networks with many layers and aids the training of networks with many layers.

Activation functions

Activation functions are applied to nodes to introduce non-linearity to our networks [35], [43]. Commonly used activation functions such as Rectified Linear Unit (ReLU) [44],

Parametric Rectified Linear Unit (PReLU) [45], TanH and Sigmoid [46], [47] enable the neural network to learn non-linear functions and control how nodes' values are propagated through the network to the following layers.

Batch normalisation layers

Batch normalisation is a method developed by Ioffe *et al.* [48] to "reduce internal covariance shift" in neural networks. Batch normalisation can be applied to hidden layers to normalise activation's $\mathbf{A}$ within the applied layer using the mean and standard deviation of the nodes in the layer, denoted as μ and σ respectively, as shown in Equation 2.15:

$$\mathbf{A}' = \frac{\mathbf{A} - \mu}{\sigma}. \tag{2.15}$$

The method introduces benefits such as improved optimisation and regularisation to neural networks. Ioffe *et al.* [48] also claim that less training time is needed if batch normalisation is applied as less careful parameter initialisation is needed within networks to obtain a trained network.

2.3.2 Training of neural networks

Neural networks are trained by providing an input to a model, which is passed through the trainable weights to an output. This output is measured against a target value, which is the desired value, using a loss function to derive how accurate the model is. Through this process, the neural network tries to approximate a function for the task at hand by mapping $\hat{y} = f(x; \theta)$, with θ being trainable parameters [35], [43]. In the remainder of this section, we discuss how parameters are optimised using gradient descent, hyperparameter optimisation and model selection.

Network optimisation

Through a process called gradient descent [35], [43], the loss function is used to adjust the weights w in the network based on the previous output's performance loss L. Each weight is updated in proportion to the negative loss gradient at the rate of the selected learning rate η, as seen in Equation 2.16:

$$\Delta w = -\eta \frac{\partial L}{\partial w}. \tag{2.16}$$

The gradient is approximated using optimisers such as Stochastic Gradient Descent (SGD) [49], [50], which uses batches of training data being passed through the network to estimate the average gradient to do weight updates with. The number of samples passed through each batch is referred to as the batch size and can play a role in the training process.

SGD performs gradient descent by randomly selecting samples, thus decreasing the computational cost. SGD has been extended to other optimisers, such as Root Mean Squared Propagation (RMSProp) [51], which adds the concept of momentum in which the learning rate is adapted based on the change in gradient, allowing for larger initial learning rates to be used. The Adam optimiser [52] extends on the RMSProp optimiser by using the second order of the gradient estimation as well as the first order of the gradient used in RMSProp.

By performing gradient descent over large quantities of data, the networks start to identify patterns in the dataset that allow for maximal performance over data samples provided in the training process. If the network has been trained sufficiently, it should be able to perform the task it was trained for on unseen data from a similar distribution.

We use different datasets in the training process to ensure that networks are trained sufficiently but not to the point that the input data is memorised. The training set is used to adjust the network weights and is typically the largest of the three sets. The validation set is used in the training process to test the network's generalisation by testing it at the end of each training set pass, known as an epoch. Network weights are never updated on the validation set, as this set gives us an insight into the performance of the

network on unseen data [35] .

Certain techniques have been developed to increase the ability of a network to perform well not only on the training dataset but on the unseen test set as well, we call these regularisation techniques. Commonly used techniques include parameter norm penalties, noise injection, weight decay, dropout and early stopping [53]. We discuss the techniques used in this study as they are applied.

Hyperparameter optimisation

Hyperparameters refer to parameters that can be altered in the optimisation process of a NN. Since many parameters can be changed within the optimisation process, such as optimisers, learning rate, and batch size, we need to find the correct combination of optimisation hyperparameters for the tasks. There are several approaches to hyperparameter optimisation which can be followed to search for hyperparameters. Grid searches, search exhaustively over every possible hyperparameter value combination within a set range by training models using the specified hyperparameters. Random grid searches [54] select hyperparameters randomly over a set range for each hyperparameter provided and can search large combinations of hyperparameters as specific values do not need to be specified. Lastly, Bayesian hyperparameter [55] searches make use of the Bayesian theorem to estimate the function of the hyperparameter space and select hyperparameter combinations to test based on previous results. The best performing hyperparameters are then selected using the validation set results of the trained network. Furthermore, optimal hyperparameters for a task may vary between architectures and datasets, and it is thus important to do hyperparameter sweeps when these factors change.

Model selection

Model selection refers to the overall process of selecting a single model from available options, taking all hyperparameters into account. Model selection takes place within the training process as well. In the starting epochs of training, losses on both the training and

validation sets should decrease, but at some point, validation loss might stop decreasing and instead increase. At this point, overtraining is taking place, and the network's generalisation on unseen data is suffering. We use a technique called early stopping to avoid overtrained networks. Early stopping selects the model that performs best on validation data in the training process instead of simply using the final network at the end of the training.

2.3.3 Adversarial neural networks

Adversarial training is a training method in which two neural networks are trained in competition with each other [4]. GAN networks have used adversarial training on many image generation and restoration tasks by adversarially enforcing the importance of specific features [6]. By applying adversarial training, networks capable of generating realistic images of humans and objects from noise vectors have been developed. In this subsection, we discuss two adversarial neural network architectures: GAN and Conditional Generative Adversarial Network (CGAN).

Generative adversarial network

A GAN [4] is a deep learning architecture used to generate data matching a desired distribution when given a random input. Figure 2.5 shows that a GAN consists of a generator and a discriminator, which compete with one another. The generator creates data similar to the original dataset based on information contained in a noise vector, while the discriminator tries to distinguish between real and generated samples from the generator network.

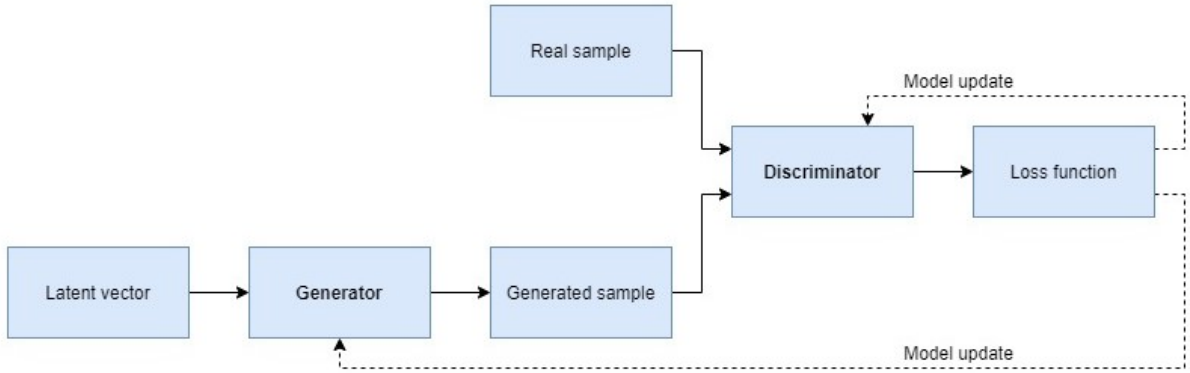


Figure 2.5: The training structure of a typical GAN model

This creates a zero-sum game between the generator and the discriminator. The generator tries to fool the discriminator, and the discriminator tries to distinguish between the generated and actual data.

Each of these networks has a loss function based on the performance of its specific task with the discriminator's loss function influencing the loss function of the generator. These two structures play this game in training, each attempting to improve its performance every epoch. Unfortunately, the layout of this training process makes it challenging to identify when the network's performance converges.

The difficulty in convergence is due to training imbalances that occur when the discriminator outperforms the generator, or vice versa, leaving the overperforming network with inconsistent feedback [6], [35]. For example, if the generator outperforms the discriminator, the discriminator is guessing whether the presented sample is real or fake, meaning the generator is training on inconsistent feedback.

Conditional generative adversarial network

CGAN [56] is a variation of the traditional GAN that makes targeted data generation possible by supplying additional inputs along with the random noise vector as part of the generator's input. This architecture was developed as data generated using GANs in their

original form only utilise random input, making the output hard to control. The solution to this problem is to add class data as input to the GAN to allow us to control what class of image is being generated.

CGAN was used by Mirza *et al.* [56] to specify which specific class should be generated after being trained on the Modified National Institute of Standards and Technology (MNIST) dataset and did so with great success. CGAN thus allows us to control the output image of the GAN to a predefined class, whereas it would otherwise generate an image from a random class.

2.4 Applications of deep learning for CSI estimation in literature

Deep learning has gained widespread success in recent years in many fields [36]–[38]. With CNNs revolutionising the computer vision field and large strides being made in NLP applications, more and more fields are researching deep learning-based models for tasks in various fields. In this subsection, we discuss why we adopt a deep learning approach to CSI estimation and investigate some work done in the field.

2.4.1 Reasons for using deep learning

CSI estimation has been implemented using statistical, machine learning, and deep learning methods in the past. We consider the complexity and the nature of the task and note that DNNs are often applied to tasks that have:

- Large amounts of labelled data available.
- Linear solutions are inadequate and better performance is required for a complex task.
- Large amounts of computing power are available for training and deployment.

CSI estimation checks two of these boxes while being challenged by the third. First, data is easy to come by since ExaBytes of data is transmitted yearly [22] and real-world conditions can be easily simulated. Secondly, as the need for spectrum efficiency rises and higher frequency transmissions are implemented, the need for increased CSI estimation accuracy rises. Statistical estimators are unable to keep up with the non-linear nature of channels, motivating the use of DNNs [57]. Unfortunately, the computational power is often limited to the user equipment in the telecommunication domain, thus making the deployment of an already trained network difficult.

This last challenge of deep learning methods for CSI is being actively researched with methods such as pruning and reducing the complexity of DNNs [58]. The ability of DNNs to outperform statistical methods is promising, as we can sacrifice some performance by reducing network sizes and attempting to challenge traditional methods in terms of performance and complexity. Thus, by increasing the performance of networks without increasing complexity, we move closer to implementing DNNs for CSI estimation tasks.

2.4.2 Related studies

Many deep learning architectures, such as MLPs, CNNs and GANs, have been applied to the CSI estimation task [59]. Specifically, architectures with convolution layers have succeeded in this task, often outperforming most statistical and machine learning methods.

The most popular approach for applying deep learning to CSI estimation is to improve an initial LS estimate [9]–[13], as is done with statistical methods such as MMSE and DFT. This approach is called super-resolution and is a popular field of study in computer vision. Super Resolution (SR) networks accept low-quality images with several levels of distortion and noise and attempt to restore the image to a higher quality version of itself [60].

An example of this method is an application proposed by Yang *et al.* [11] that takes calculated LS estimation as input and uses a DNN model to generate a channel estimation based on the previous channel state and new LS calculations. Furthermore, CHANNELNET [9] makes use of a super-resolution network consisting of SR CNN [61] and denoising

CNN [62] that takes pilot symbols as input to generate CSI. Both of these methods outperformed traditional LS and MMSE methods, demonstrating the validity of deep learning methods in this field.

With these SR networks providing good results on the CSI estimation task, we look to architecture that further improves SR results in the computer vision field. Super Resolution Generative Adversarial Network (SRGAN) is an adversarial architecture proposed by Ledig *et al.* [5], that introduces adversarial training methods to the SR task. The core concept of this architecture is that rather than optimising the MSE loss between pixels, a discriminator is introduced to evaluate the generator, based on extracted features from images. Ledig *et al.* [5] claim that this increases the accuracy of high-frequency features in images.

The work by Ledig *et al.* [5] is expanded to the CSI estimation task by Zhao *et al.* [16] adapting the network to perform SR on a provided LS estimation. The SRGAN adapted for OFDM CSI estimation also employs adversarial training to create a CNN architecture with improved results due to the addition of a discriminator in the training process.

2.5 Adversarial architectures used in this study

In this study, we use the original SRGAN architecture as well as the SRGAN architecture proposed by Zhao *et al.* [16] for CSI estimation tasks. We refer to the model proposed by Zhao *et al.* [16] as adversarial ResNet moving forward.

2.5.1 SRGAN

SRGAN [5] is a GAN designed for image SR tasks, specifically image upsampling tasks. The adversarial training used by Ledig *et al.* [5] for SRGAN is reported to push the generated SR images closer to the original images' manifold. This SR implementation results in a higher SNR and sharper, high-frequency feature generation. SRGAN achieves these

results by moving away from using the MSE to compare individual pixels to each other. Instead, SRGAN compares feature maps of the images obtained by a Visual Geometry Group (VGG) network [63]. Applying this VGG loss $l_{content}$ to images generated, the Euclidean distance is measured between the feature maps of real and generated images. The loss function, described in Equation 2.17, averages the MSE loss between the feature maps, generated by the VGG-19 network ϕ, of the high-resolution image I^{HR} and the image generated from low resolution using the generator $G(I^{LR})$:

$$l_{content} = \frac{1}{ij}\sum_{y=1}^{i}\sum_{x=1}^{j}(\phi(I^{HR})_{x,y} - \phi(G(I^{LR}))_{x,y})^2. \tag{2.17}$$

Here i and j represent the respective dimensions of the feature maps. The content loss forms part of the final loss function l_{Gen} in Equation 2.19 by being added to the adversarial loss used to evaluate the generator using the discriminator $l_{adversarial}$, shown in Equation 2.18.

$$l_{adversarial} = -log(D(G(I^{LR}))) \tag{2.18}$$

$$l_{Gen} = l_{content} + 1e^{-3} * l_{adversarial} \tag{2.19}$$

SRGAN's generator consists of a convolution layer using a PReLU [45] activation function followed by several residual blocks and another convolution layer, all using batch normalisation [48]. Two 2× upsampler layers are added to upscale the physical dimensions of the feature maps before feeding the maps to an output convolution layer. All residual blocks use 64 channels before these are increased to 256 channels in the upscaling blocks.

The discriminator's structure closely follows a VGG network with eight convolution layers. Each of these layers reduces the feature maps' size while increasing the number of channels until the output is passed to a 1 024 node dense layer followed by a single node layer connected to a sigmoid activation function. This sigmoid output is then used with Binary Cross-Entropy (BCE) loss to determine whether samples are real or generated. The discriminator is trained using the BCE loss function and passing both real and generated data samples to the loss function as seen in Equation 2.20:

$$l_{Disc} = \frac{-log(1 - D(G(I^{LR}))) - log(D(I^{HR}))}{2}. \tag{2.20}$$

2.5.2 Adversarial ResNet

Zhao *et al.* [16] drew inspiration from the use of adversarial training in SRGAN and adapted the architecture to be suitable for CSI estimation of OFDM-based transmissions.

This adapted architecture receives LS CSI estimations as inputs to the generator G. The generator consists of several layers of residual blocks that utilise skip connections to alleviate disappearing gradient problems often found in DNNs. The generator's output is measured to the target CSI using MSE loss between corresponding CSI symbols. Adversarial training is applied by the discriminator D, which is trained in the same way as for SRGAN, depicted in Figure 2.6. The network attempts to classify the generated sample with its output measured by a BCE loss function. Finally, we examine the MSE loss function $l_{content}$ in Equation 2.21:

$$l_{content} = \frac{1}{ij} \sum_{y=1}^{i} \sum_{x=1}^{j} (H_{x,y} - G(\hat{H}_{Ls})_{x,y})^2. \tag{2.21}$$

Here i and j represent the respective dimensions of the CSI samples with H being the real CSI and $\hat{H}$ the CSI estimate. The $l_{content}$ forms part of the final loss function l_{Gen} in Equation 2.19 by being added to the $l_{adversarial}$ loss function of the discriminator in Equation 2.18.

Compared to the original SRGAN, Zhao *et al.*'s [16] implementation removes the use of feature extractors and implements a pre-upsampling strategy that removes the upsampling layers in the generator. Zhao *et al.* [16] obtain results competitive with state-of-the-art statistical methods in a SISO environment attributing the success to these architecture changes and adversarial training. The adapted SRGAN architecture can be found in Figure 2.6.

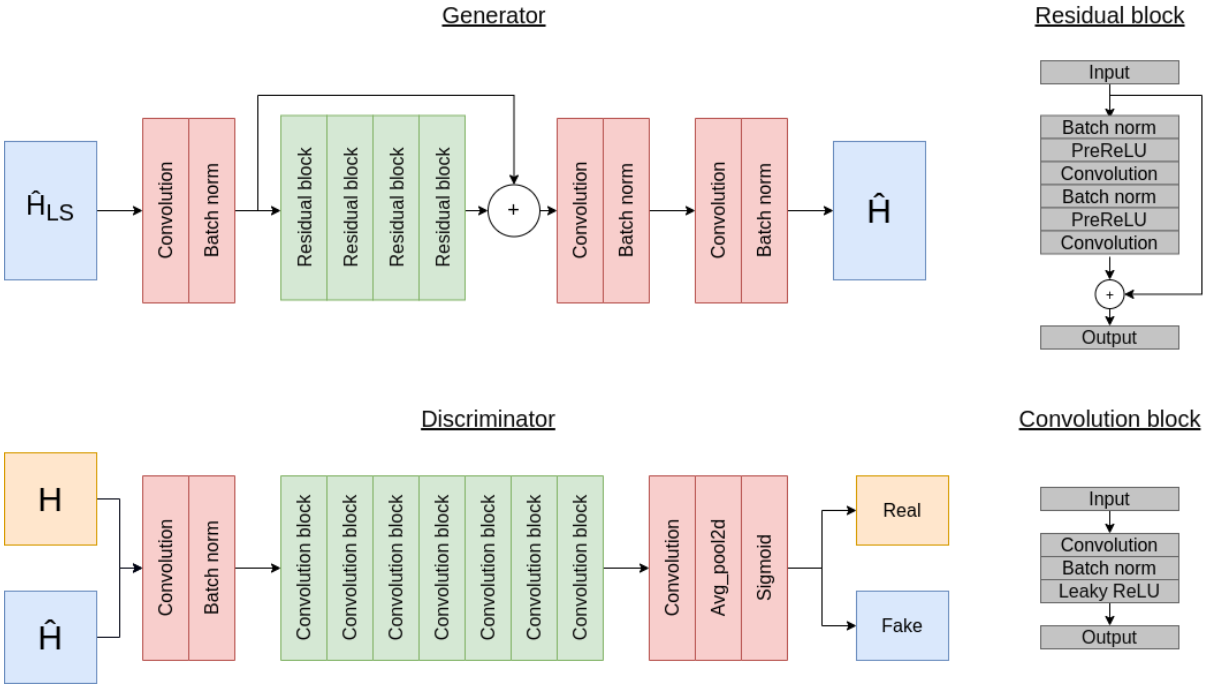


Figure 2.6: Illustration of the modified SRGAN network structure applied by Zhao *et al.* [16].

2.6 Conclusion

This chapter discusses concepts of interest to the study of wireless communication systems, CSI estimation, and deep learning. We motivate using the LTE standard for the wireless communication system and why DNNs are a suitable tool for the CSI estimation task. After describing related studies, we describe two adversarial architectures suitable for our work: one that has been applied to this domain, and one that has not.

Chapter 3

Wireless system simulation

In this chapter, we describe the simulation constructed in Matlab® to simulate a wireless LTE communications system. This physical layer simulation is used for testing and dataset generation in the rest of the study.

3.1 Introduction

By using the LTE toolbox available in Matlab®[1], we can simulate the transmission of OFDM symbols over wireless channels that represent real-world conditions. We create this simulation to assist us with:

1. Generating datasets tunable to different channel parameters and modulation schemes.
2. Creating baselines of frequently used methods for CSI estimation for both SISO and MA environments.
3. Simulating the performance of neural networks in complex MA environments.

[1] `https://www.mathworks.com/products/lte.html`

By creating a simulation, we allow for the creation of reproducible datasets and experiments using random number generator seed selection. The creation of this simulation is further motivated by the need to closely inspect aspects of the wireless systems under various conditions.

In the following sections, we discuss the necessary building blocks of physical layer systems and how they are implemented into the simulation. First, we discuss physical layer systems as a whole, followed by a description of all building blocks and figures collected from the simulation to aid understanding of how channel impairments affect CSI.

3.2 Physical layer simulations

Communication systems are developed using the standard seven-layered Open Systems Interconnection (OSI) model [64]. Each layer in this model plays a different role in the communication process. The objective of the physical layer in the OSI model is to move bits from a transmitter to the receiver over a medium, either a physical cable or a wireless channel. The physical layer is likewise responsible for preparing the bits received from the data link layer for transportation over the medium. This process in wireless systems usually entails employing modulation schemes to transform bits into QI symbols. After the QI symbols have been transmitted, the physical layer at the receiver side is responsible for demodulating the symbols and passing on bits to the receiver's data link layer for further processing.

We mimic the LTE physical layer depicted in Figure 3.1 for the simulation design in this chapter. The content of this figure is obtained from literature [17], [65] and is consistent with the physical layer elements described in the LTE standard. The first block of this physical layer simulation generates bits which are modulated to QI symbols using QAM modulation, as QAM is the modulation technique used in LTE downlink transmissions. Next, the QAM symbols are used to populate the resource blocks, along with the pilot symbols. These resource blocks are further modulated to OFDM modulation and transmitted over the multipath channel containing simulated delay profiles and Doppler

spread.

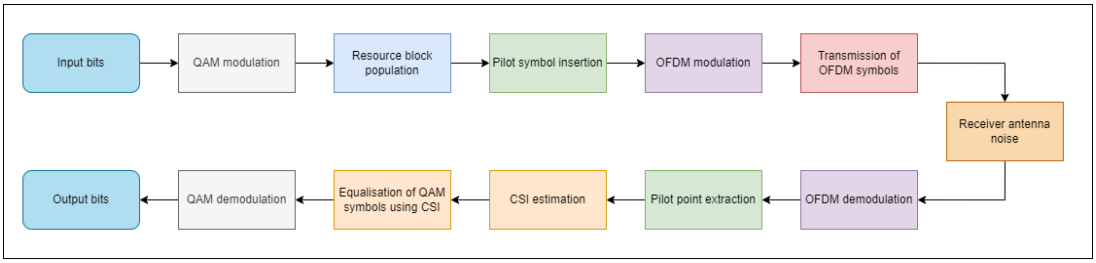


Figure 3.1: Physical layer blocks implemented to create the wireless communication simulation

Before demodulation, OFDM symbols transmitted over the simulation channel are subjected to receiver antenna noise at a preselected SNR level. Pilot symbols are extracted from the received resource grid produced by the demodulated OFDM symbols. Finally, CSIs estimation and equalisation are performed over the resource grid before demodulating the QAM symbols back to bits.

3.3 Resource grid construction

Modulation plays an important role in transmitting bits between two devices as accurately as possible without sacrificing speed or bandwidth usage. In the simulation, we make use of two types of modulation schemes: QAM, and OFDM modulation. The first is used to convert bits to QI symbols while the second enables the effective use of the available bandwidth.

In order to apply OFDM modulation we need to create a 2-dimensional grid containing one QAM-modulated symbol per subcarrier timeslot. In Figure 3.2 we see that some of the spaces in this grid are reserved for pilot symbols, which are placed at specific subcarriers known to both the transmitter and receiver and decided by the LTE standard.

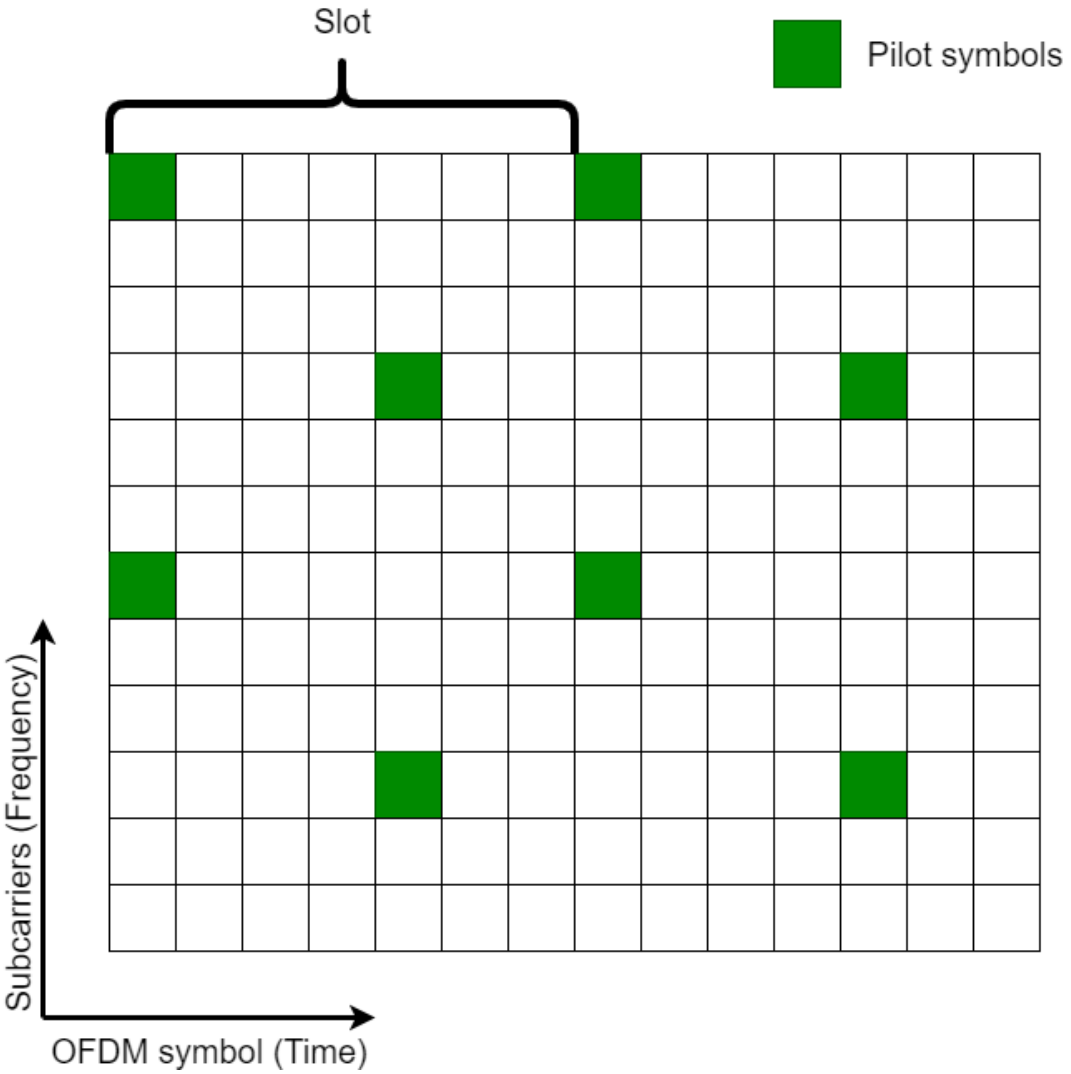


Figure 3.2: A representation of a subframe with pilot positions shown in green

This populated resource grid is used to create 14 OFDM multiplexed symbols. The resource block is thus multiplexed into 14 OFDM symbols with 600 subcarrier symbols per OFDM symbol for a 10 MHz transmission. According to the LTE standard, we insert pilot points in both the time and frequency domain, as indicated in Figure 3.2. Placing pilot symbols in the positions indicated in Table 3.1 allows us to estimate channel impairments in both the time and frequency domains.

Table 3.1: Parameters used in LTE physical layer simulation for modulation and resource block construction.

Parameter	Value
Pilot insertion time	1, 6, 8, 12
Pilot insertion frequency	Every 6th subcarrier
QAM modulation order	4 and 16
Subcarrier spacing	15 KHz
Cyclix prefix	4.7 ms
Number of subframes	50
Channel bandwidth	10 MHz

Table 3.1 further depicts the parameters used for the simulation's modulation, resource grid population and pilot point insertion subcarriers. The LTE standard prescribes pilot point values, subcarrier spacing and cyclic prefixes. We decide to use 4-QAM and 16-QAM in this study as they provide constellation diagrams that are less dense than large-order QAM modulation maps and thus easier to analyse. The number of subframes in a resource block is selected as 50, which generates a resource grid with a width of 14 and a depth of 600. This number generates an OFDM symbol with a bandwidth of 10 MHz, which is one of the bandwidths used in the LTE standard.

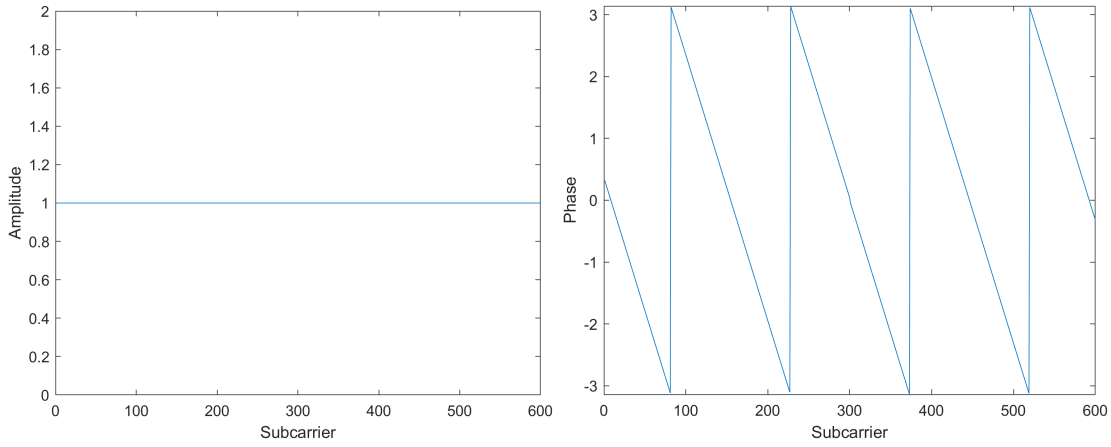
3.4 Channel impairments

Communication systems are measured by their ability to move data from one point to another. The biggest obstacle to this objective is the channel over which data is transmitted, which impairs the quality of the received data, making retrieving accurate data difficult.

The channel used in the simulation is an implementation of the LTE toolbox multipath channel found in Matlab®. This channel enables us to simulate impairments such as multipath effects, delay profiles and Doppler spread on transmitted data. In addition, the ability to select parameter values for each of these impairments allows us to create datasets of many different LTE environments.

3.4.1 Simple channel

We start by investigating the CSI of a channel that applies no impairments to the transmitted data. To investigate the change in CSI, we compare the amplitude and phase of the QAM modulated symbols before and after being transmitted over the channel in both the time and frequency domains. Observing the amplitude and phase changes in Figure 3.3, we note that the CSI amplitude remains unchanged at one, meaning no attenuation has occurred. However, the phase of the CSI changes, indicating that the equalised QAM symbols rotate in a unit circle on the constellation diagram.



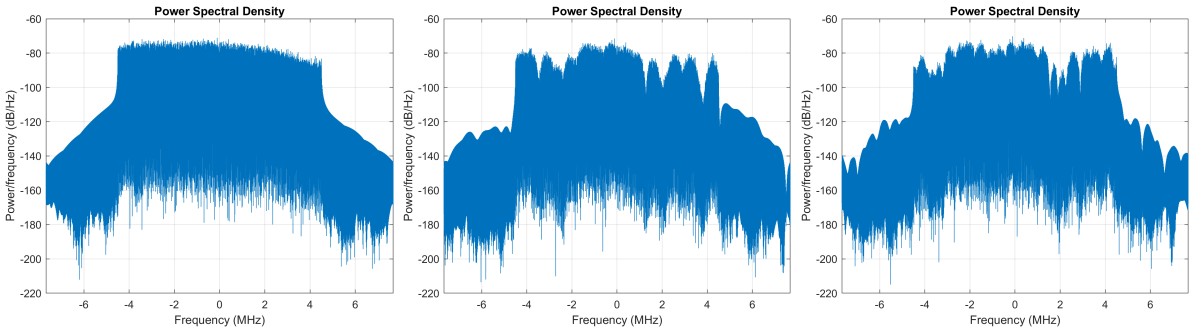
(a) Amplitude (b) Phase

Figure 3.3: Figures depicting the amplitude and phase shifts of CSI in the frequency domain for a simple channel with no impairments.

Observing amplitude and phase in the time domain for a single subcarrier shows no change over OFDM symbols, meaning that the CSI does not change over time. This change in phase is caused by a timing offset when demodulating the OFDM symbols. This problem can be alleviated by offsetting the frame before demodulation, but will not be done for this study as this phase shift can be seen as a simple channel condition to test models on.

3.4.2 Delay profiles

Since we observed that a simple channel only affects the phase component of received symbols, we investigate the effects of power delay profiles. Delay profiles add attenuation to signals over different propagation paths. The LTE standard specifies three ITU delay profiles used in simulation and testing: EPA, EVA and ETU as described in Section 2.2.3. Each of these delay profiles is more disruptive to the transmitted signal than the previous one. Delay profiles affect the OFDM transmissions in the frequency domain. We observe this in the power density plots of transmitted OFDM symbols under different delay profile conditions displayed in Figure 3.4. The impairment caused by the delay profiles can be measured by the frequency and severity of attenuation in power in the frequency domain.

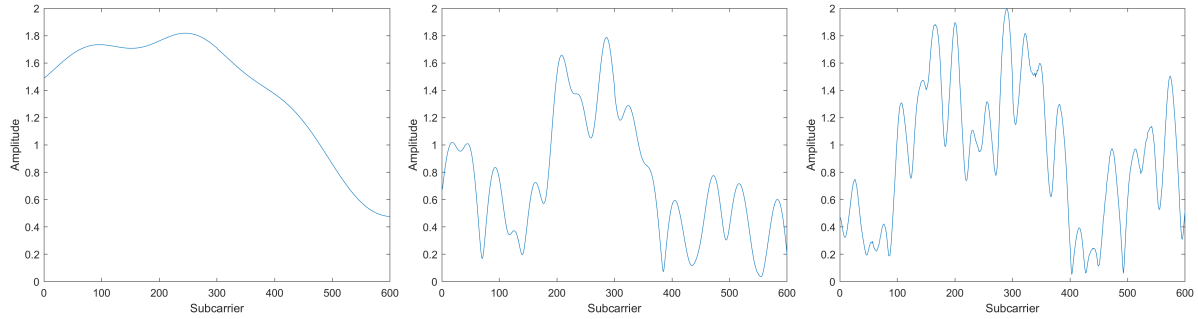


(a) EPA delay profile (b) EVA delay profile (c) ETU delay profile

Figure 3.4: Power spectral density plots depicting the effect delay profiles have on transmitted OFDM symbols in the frequency domain.

We can further investigate the impairment caused by delay profiles by investigating the amplitude variation of CSI in the frequency domain. Figure 3.5 shows that each delay profile changes the amplitude of the received symbols at different rates, thus making it increasingly difficult to estimate accurate CSI.

Similar trends are observed for phase changes in the frequency domain, with phase changes occurring at faster rates as delay profiles progress. We also observe CSI changes in the frequency domain but find that no variation over time occurs when delay profiles are applied to the channel.

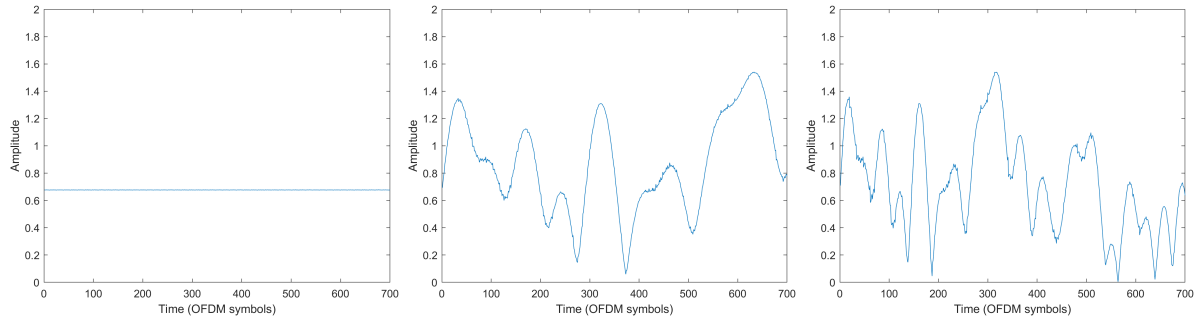


(a) EPA delay profile (b) EVA delay profile (c) ETU delay profile

Figure 3.5: Figures depicting the amplitude shifts of CSI in the frequency domain for multipath channels with different delay profiles.

3.4.3 Doppler spread

In order to analyse the effect of Doppler spread, we note that it takes effect in the time domain due to the relative movement between the transmitter and receiver. Knowing this, we observe the variation of CSI amplitude over time for a channel with an EVA delay profile and a set amount of Doppler spread in Figure 3.6.



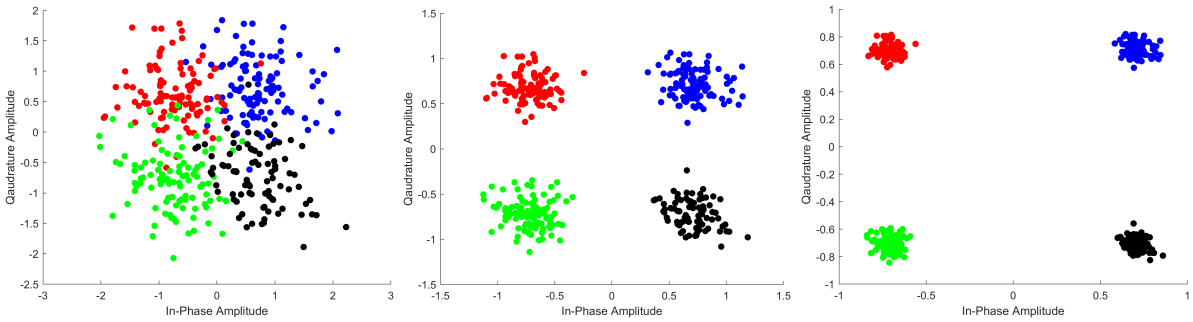
(a) No Doppler spread (b) 100 Hz Doppler spread (c) 200 Hz Doppler spread

Figure 3.6: Figures depicting the amplitude shifts of CSI in the time domain for a multipath channel EVA delay profile under different Doppler spreads.

From these results, we see that the amplitude change brought on by the delay profile stays constant when the channel applies no Doppler spread. Change in amplitude is only observed when Doppler spread is applied. The amount of Doppler spread dictates the rate of change, as can be observed when comparing a channel with 100 Hz Doppler spread to one with 200 Hz Doppler spread.

3.4.4 Antenna noise and interference

After transmission over the channel, we apply AWGN to the OFDM symbol before demodulating. This noise is added to simulate the combined effect of all noise-inducing factors such as receiver antenna noise, electrical equipment in the channel and heat-induced component noise. We simulate the amount of noise using SNR and can observe the noise on the QI plot of QAM symbols, as displayed in Figure 3.7. The SNR can range from mild inconvenience, as shown by the 20 dB figure, to completely debilitating at lower SNRs, especially when paired with impaired channel conditions.



(a) 0 dB SNR (b) 10 dB SNR (c) 20 dB SNR

Figure 3.7: 4-QAM modulated QI points after an OFDM symbol has been subjected to various SNRs of AWGN. The colours signify the bits represented by the specific QI symbol.

MA environments are created by utilising a different channel for each antenna to antenna connection. This therefore means that an environment with one transmitter and two receivers will have two channels with a set amount of correlation between CSI. We usually assume that antennas are spaced an adequate distance apart to ensure independent channels. We have four channels for an environment with two transmitters and two receivers. MA environments thus add additional interference in multitransmitter setups as two different signals arrive at a single receiving antenna. This interference makes decoding only one of the antenna symbols difficult as the QI symbols superimpose in both the time and frequency domains.

3.5 Channel estimation using LS and LTE-MMSE

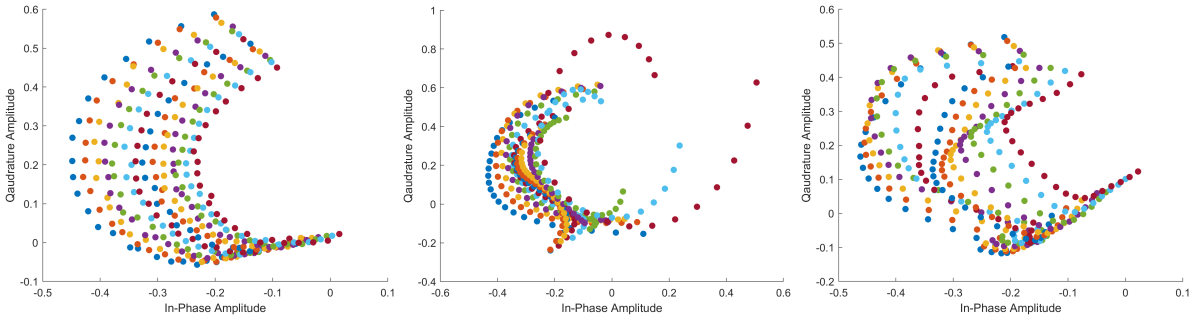
CSI estimation and equalisation play a significant role in communication systems as they estimate impairments brought on by channels and equalise received data using these estimations. In this study, we use CSI estimation techniques that require only first-order information obtained from pilot symbol data. We do not use techniques like MMSE and Linear Minimum Means Squared Error (LMMSE) that need second-order statistics of the channel, as this increases the system's complexity and is rarely used in practice. For this study, we use LS and LTE-MMSE as CSI estimators. We use these techniques as baselines for comparison with the neural network estimators that only use CSI obtained from pilot symbols to make estimations.

We use the method described in Section 2.2.4 to estimate CSI in the simulation for the LS CSI estimator. The implementation first creates an estimation for the subcarriers in the four OFDM symbols that contain pilot symbols over the entire OFDM symbol. These four symbols are then interpolated using spline interpolation over the rest of the subcarriers in the time domain, creating a bicubic interpolation. In this study we implement a single LS estimation over the entire resource grid in order to include as many pilot symbols as possible in the estimation.

For the LTE-MMSE CSI estimator, we make use of the LTE toolbox method provided in Matlab®. The LTE-MMSE CSI estimation is a two-step process that entails doing a downlink channel estimation and applying the estimation using a simplified MMSE equalisation method over the entire resource grid. The downlink method uses several pilots' averaging methods to reduce the noise on the pilot symbols. In this process, an estimated noise metric is also calculated. In addition, the downlink method uses virtual pilots created from real pilot data to help provide more pilot symbols to the denoising process. After this estimation has been completed, the received data, channel estimation and noise estimation are passed to the simplified MMSE method. This LTE-MMSE method equalises the received data with the channel estimate and applies an inverse noise matrix to reduce noise. From this point forward we refer to this simplified MMSE simply

as ‘MMSE’.

We model a channel using an EVA delay profile, 100 Hz of Doppler spread and 20 dB receiver noise to observe how LS and MMSE estimators differ in performance. Then, this channel’s true CSI is calculated and estimated using CSI estimators. Finally, we plot the QI points of the calculated and estimated CSI in Figure 3.8. Each QI point refers to a corresponding received QAM symbol and how it was affected due to channel impairments, while each colour represents an individual OFDM symbol of the plotted subframe. If a symbol has undergone no impairments, the CSI estimation will be one.



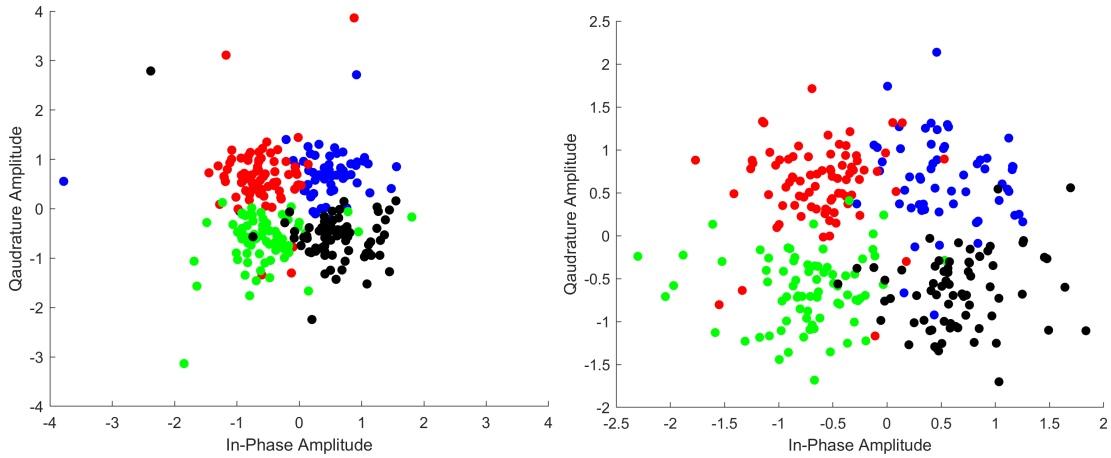
(a) Perfect CSI (b) LS (c) MMSE

Figure 3.8: QI plots of CSI obtained using different estimation methods. Each colour represents an individual OFDM symbol of the plotted subframe.

From merely observing Figure 3.8 we can gather that MMSE creates much smoother estimates than the LS estimate, which produces inconsistent estimates with many outliers. Comparing both estimates to the real CSI, we see that the MMSE estimator provides more consistent results between OFDM symbols than the LS estimator.

To further illustrate the effect of estimation and equalisation, we equalise received OFDM demodulated QAM symbols using CSI. We equalise 4-QAM data sent over the same channel using LS and MMSE methods. In Figure 3.9 we observe that both methods provide adequate results by adjusting the received symbols close enough to the demodulation point to retrieve most of the transmitted bits correctly. However, MMSE places the points more accurately and would thus perform better in higher-order QAM modulations and channels with increased impairments. The LS estimation also includes an equalised symbol that has been severely misplaced due to the erroneous estimation of CSI and has

therefore been omitted in the figure.



(a) Demodulated QAM symbols after being equalised with LS CSI.

(b) Demodulated QAM symbols after being equalised with MMSE CSI.

Figure 3.9: Demodulated 4-QAM symbols after being equalised with estimated CSI. The colours indicate the bits represented by the specific QI point.

3.6 Multi-antenna environments

Wireless systems have long passed the time of using only a single antenna to transmit data. Instead, multiple antennas are used to increase a communications system's throughput. In this study, we examine single and MA environments, focusing on antenna diversity specifically for MA environments. Since MA environments have multiple channel states to estimate, we expect to see that the quality of CSI estimation is an essential aspect of BER performance.

We implement receiver diversity using the EGC method discussed in Section 2.2.5, as it requires no additional noise estimates while still portraying the advantages of using antenna diversity. In Figure 3.10 we implement EGC using perfect CSI and EPA delay profile with no Doppler spread to showcase the advantage gained by using receiver diversity over multiple channels. The figure shows the decrease in BER obtained compared with SISO channels when implementing receiver diversity.

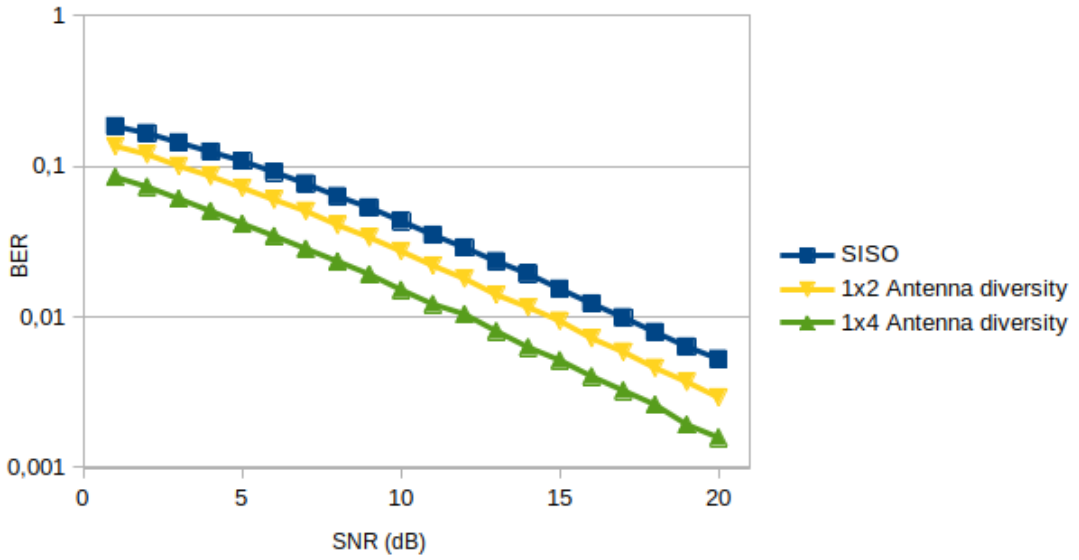


Figure 3.10: BER of a receiver diversity system with EPA delay profile depicting the average SISO channel performance of the system along with the performance of two and four receiver antennas.

For transmitter diversity, an Alamouti code is implemented to create OSTBC to be transmitted over two transmitter antennas. To investigate the BER of this method using perfect CSI we need to obtain each channel's CSI individually, as the interference caused by two OFDM symbols being received simultaneously makes calculating a single channel's CSI impossible. To overcome this challenge, we make use of an identical channel environment to transmit from only one of the two transmitting antennas at a time. The CSI obtained from the channel twin is then applied to the originally transmitted data. Furthermore, we only follow this method for perfect CSI, as CSI estimators do not transmit on certain subcarriers in transmitter diversity environments to enable other transmitting antennas to transmit pilot symbols without interference.

We implement the Alamouti method in the simulation for both single receiver and multireceiver environments. Figure 3.11 shows the BER results of the implementation using perfect CSI and EPA delay profile with no Doppler spread. From these results, we can see that having both transmitter and receiver diversity in conjunction with an Alamouti code implementation provides the best BER.

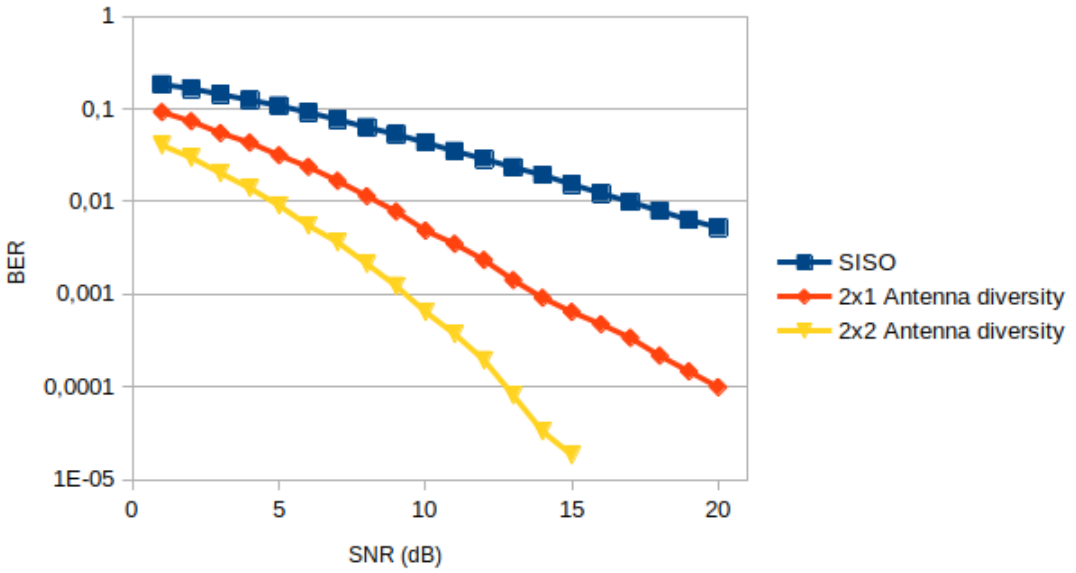


Figure 3.11: BER of a transmitter diversity system with EPA delay profile depicting the average SISO channel performance of the system along with the performance of one and two receiver antennas.

3.7 Simulation results

With all blocks of the physical layer simulation developed, we now combine these blocks as described in Figure 3.1. This simulation is used for both baseline and data generation throughout the rest of the study. To better understand neural networks' expected performance, we create BER vs SNR plots of different antenna environments, channel impairments, CSI estimators and modulation schemes. In Appendix A.1, we compare the results of this simulation to other work and find that the BER results are within a 95% confidence interval of other physical layer simulations.

We draw these preliminary baselines using a multipath channel that applies an EVA delay profile and 100 Hz Doppler spread to the transmitted symbols. To ensure consistent and reliable results we transmit 20 resource grids (14 OFDM symbols with 600 subcarriers each) per SNR over the SNR range of 1 dB to 20 dB. All baselines are also generated using both 4-QAM and 16-QAM, as both are used throughout this study.

Plotting the BER curves of a SISO system using our CSI estimation methods in Figure 3.12 confirms that MMSE outperforms LS as was expected from the observations earlier in Section 3.5. However, a decrease in BER is observed when increasing the modulation order from 4-QAM to 16-QAM, confirming that higher modulation orders require more accurate CSI predictions to obtain the same BER as lower order modulation schemes.

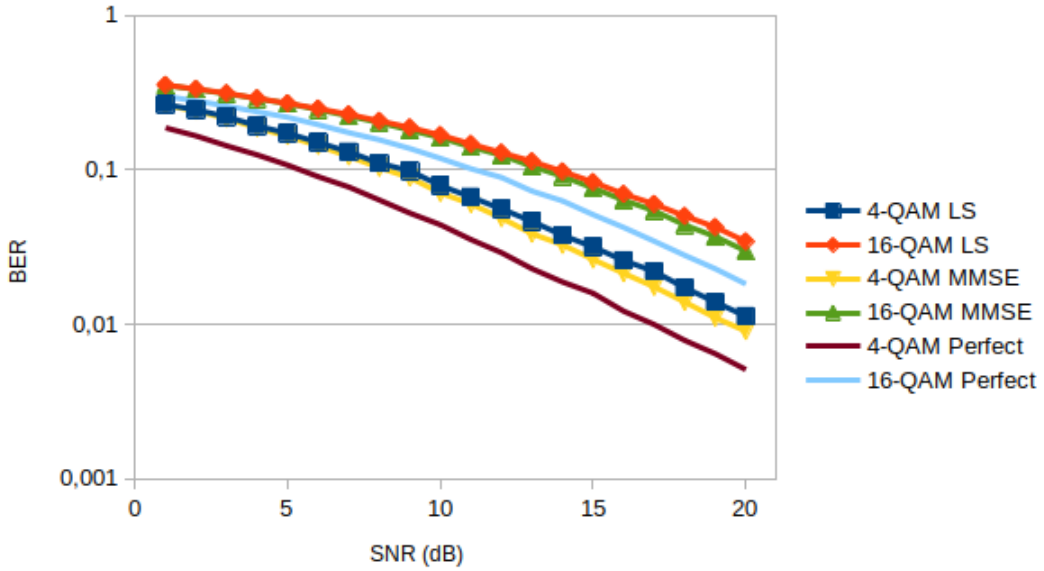


Figure 3.12: The BER of a SISO system with an EVA delay profile and 100 Hz Doppler spread. We depict the performance of 4 and 16-QAM modulations using LS, MMSE and perfect CSI estimations.

We observe the same trend between modulation orders when examining the BER of MA environments in Figures 3.13 and 3.14. In these figures, we further see that MMSE outperforms LS, especially at higher SNR levels.

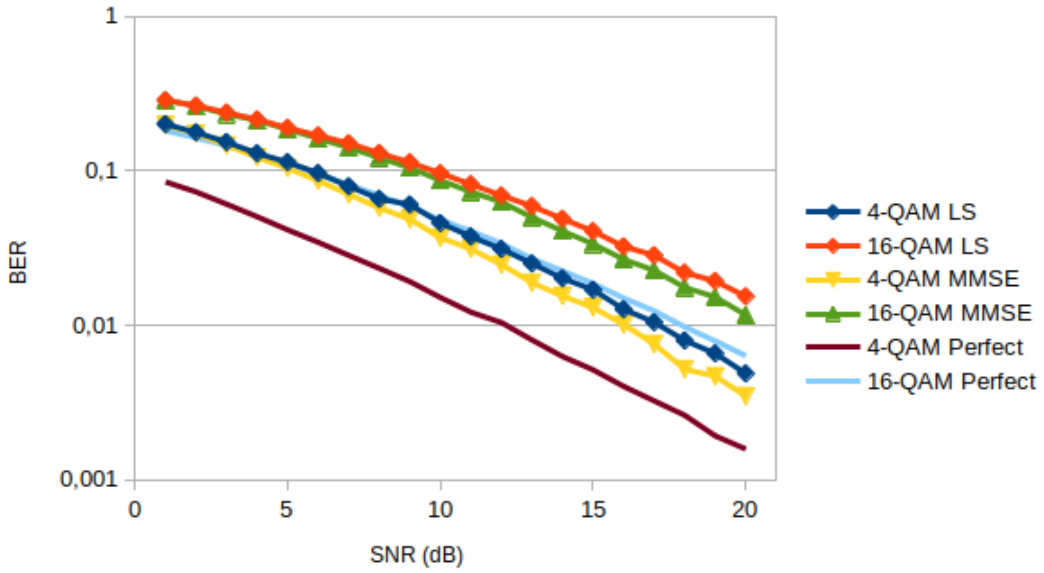


Figure 3.13: The BER of a four antenna receiver diversity system with an EVA delay profile and 100 Hz Doppler spread. We depict the performance of 4 and 16-QAM modulations using LS, MMSE and perfect CSI estimations.

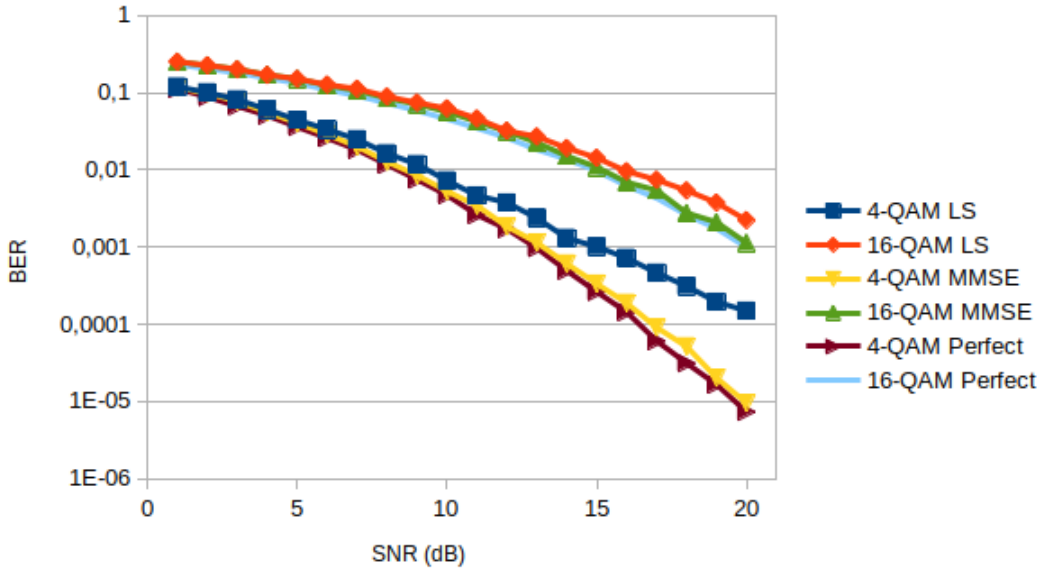
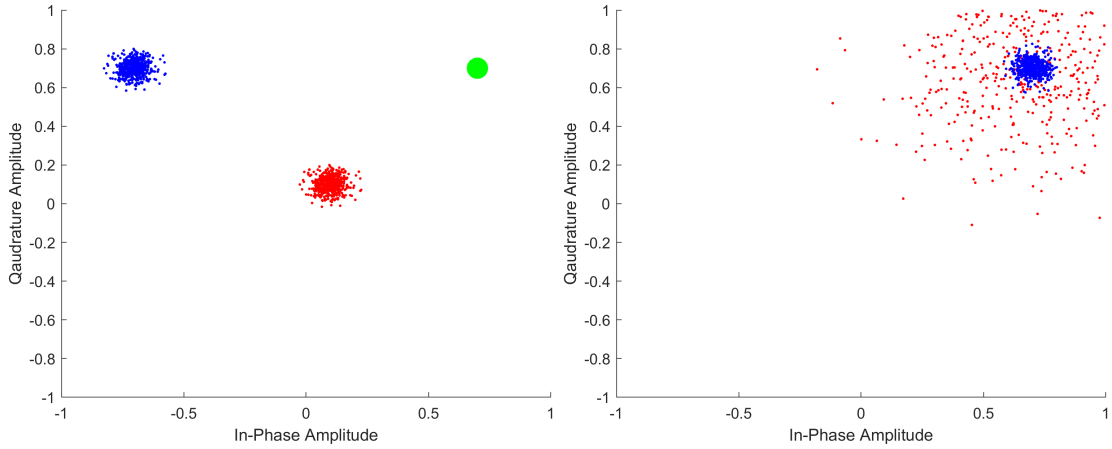


Figure 3.14: The BER of a two receiver antenna transmitter diversity system with an EVA delay profile and 100 Hz Doppler spread. We depict the performance of 4 and 16-QAM modulations using LS, MMSE and perfect CSI estimations.

The perfect CSI curve acts as a theoretical limit for methods with no noise reduction and expresses the maximum performance obtainable if channel conditions are estimated perfectly. The SISO and receiver diversity results in Figures 3.12 and 3.13 perform much worse than the perfect CSI, but the Alamouti implementation is very competitive with this curve.

Note that we do not achieve a BER of zero in the simulation results. Perfect equalisation is not possible as noise is added after the channel impairments and is assumed to be impossible to estimate due to its random nature. Furthermore, the theoretical BER curves are not identical for all channel conditions; this is because noise is added before equalisation can take place and scales along with the symbols being equalised. The Alamouti implementation does not suffer from this problem due to the nature of the method.

We demonstrate the noise enhancement problem in Figure 3.15 where the same amount of noise is added to two different symbols under different amounts of attenuation. When equalising the received symbols using perfect CSI, we see that the area of error increases in the instance of the symbol that underwent attenuation but stays the same for the symbol that only underwent a phase shift but not attenuation. We thus have two perfectly equalised symbols, but one performs worse than the other due to the attenuation experienced before AWGN was added.



(a) Before equalisation (b) After equalisation

Figure 3.15: Here we depict the noise enhancement problem. Subfigure a shows two CSI conditions in red and blue, with green being their original position. Subfigure b shows the result of equalisation on AWGN when perfect CSI is applied.

3.8 Conclusion

In this chapter, we discussed the physical layer simulation used in this study to generate baselines and datasets to train and compare our neural networks. In this process, we discuss and examine how resource grids are populated, what channel impairments are added to the transmitted data and how each affects the transmitted data in both the time and frequency domains. CSI estimators, how they are implemented, and the expected performance, are also discussed. The implementation of MA environments is investigated along with BER curves to show the possible performance in each case. Finally, the entire simulation is brought together to show different modulation schemes and CSI estimators' effects when implemented in different environments. We also discuss the noise enhancement problem, which makes perfect equalisation difficult.

With the simulation complete and a better understanding of the CSI data established, we can start training neural networks for CSI estimation on datasets created with this simulation.

Chapter 4

CSI estimation using MLPs

We investigate feature representations and hyperparameter choices using MLPs to better understand how CSI data interacts with deep learning methods.

4.1 Introduction

In this chapter, we investigate the use of MLPs for channel state estimation. MLPs provide a straightforward initial model to explore the training data. We start by testing different feature representations for training neural networks while simultaneously optimising hyperparameters for the task. We evaluate these factors by training MLPs, testing their ability to generate CSIs, and benchmarking results against LS and MMSE estimations. Hyperparameter searches are explained in detail and used to make decisions that lead to training and selecting the best performing networks. In this optimisation process, we investigate the use of batch normalisation and its interaction with Euclidean and non-Euclidean data representations. By the end of this chapter, we aim to have selected a feature representation to use in this study and to understand the roles that hyperparameters play in the optimisation process. Note that sections of this chapter has been

published in Oosthuizen *et al.* [20].

4.2 Architecture

We use an MLP for the architecture in this chapter as typical MLPs consist of fewer trainable parameters and hyperparameters than CNNs and GANs. Furthermore, the simplicity of the MLP makes training and optimising the network straightforward and easy to analyse.

The MLP architecture used in this chapter has a rectangular architecture comprised of a set number of fully connected layers, each containing the same number of neurons. Additionally, we include batch normalisation layers, discussed in Section 2.3.1, in the MLPs to stabilise and speed up the training process. This combination of trainable parameters, non-linearity and training aids enables the MLP to perform complex regression tasks such as CSI estimation. The MLP used in this chapter uses a TanH activation function, which has previously been used in CSI estimation tasks [66].

4.3 Dataset

We use the simulation discussed in Chapter 3 to generate datasets for the MLPs to be trained and evaluated on. The data samples used for the datasets in this section consist of 7 OFDM symbols, each containing 12 subcarriers i.e. a slot as presented in Figure 4.1. Subcarriers are filled with QI symbols modulated with 4-QAM modulated symbols. Slots are thus used for samples as they are the smallest data frame available in the LTE standard, discussed in Section 2.2.3. Starting with slots as data samples and MLPs allows us to investigate how neural networks interact with the datasets.

We depict the data flow of the training and evaluating process in Figure 4.1. The first step is pre-processing the slots by extracting the four pilot symbols from the data sample. We calculate the CSI of these four pilot symbols to use in the project. Feature extraction

must occur on these CSI symbols as neural networks have difficulty working with complex numbers such as QI points. Therefore, we extract two features from each QI symbol (how and which features are extracted is described in more detail in Section 4.5.1) and place the total of 8 features into a flattened tensor. The 168-wide tensor generated from the MLP output is used as the target of the MLP. This 168-wide target tensor is finally recombined using the features to create the slot's QI CSI symbols. The objective of the training is thus to predict the entire slot's CSI (12x7=84 in this specific case) from the four pilot symbols.

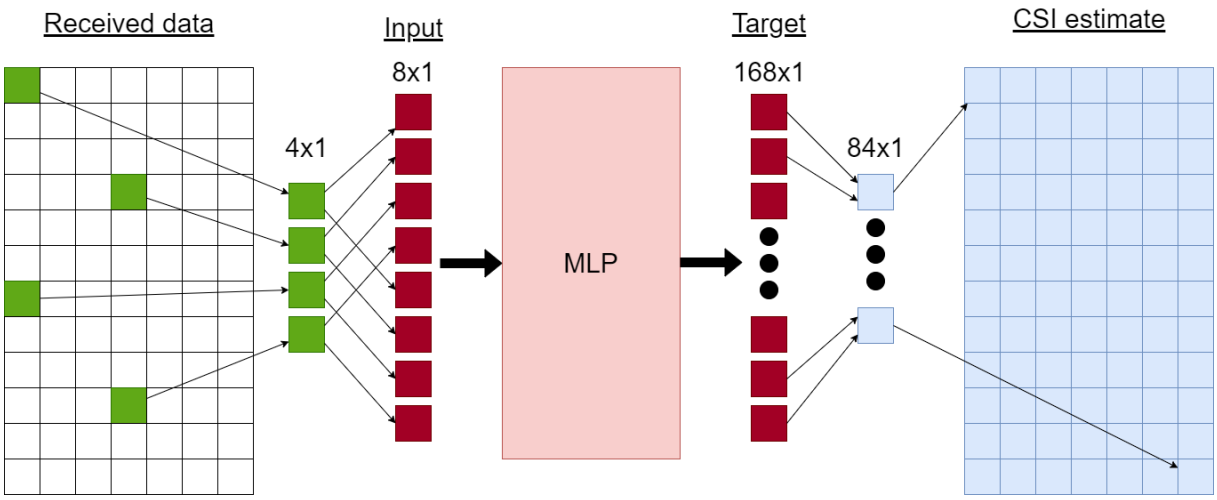


Figure 4.1: The process of creating features as input for the MLP can be seen on the left of this figure, while CSI reconstruction from the MLP's output is illustrated on the right.

To analyse the ability of the MLP under different conditions, we create several datasets for the MLPs to train on. This chapter refers to the datasets as 'Noiseless', 'Noisy', 'Delay profiles' and 'Doppler spread'. Each dataset increases the number of impairments transmitted data experiences over the simulated channel. We list the impairments of each dataset in Table 4.1. These impairments are implemented as discussed in Section 3.4.

Table 4.1: Channel conditions used in the simulation for each dataset generated in this chapter.

Dataset name	Channel conditions
Noiseless	Simple channel
Noisy	Noiseless dataset with AWGN (20 dB SNR)
Delay profiles	Noisy dataset and one of the three delay profiles
Doppler spread	Noisy dataset, EVA delay profile and different amounts of Doppler spread

Each dataset contains a training set, a validation set, and a test set of 20 000, 5 000, and 5 000 samples, respectively. Along with this, we generate LS and MMSE baselines from the simulation discussed in Chapter 3 to measure the performance of the MLP.

4.4 Training protocol

To ensure that the best hyperparameters are chosen for the architecture and training, we use a hyperparameter grid search in the training protocol. We use Adam as an optimiser and MSE as a loss function in this chapter, as discussed in Section 2.3.2. Early stopping is implemented to obtain the best performing network from the training process.

The hyperparameter grid search sweeps over network depth, width, batch size and learning rate. We refer to network width and depth as architectural hyperparameters, and batch size and learning rate as training hyperparameters. All hyperparameter grid searches are performed over three initialisation seeds and averaged to ensure that strong or weak weight initialisation do not affect the trend analysis. All loss and BER values presented are thus an average of three hyperparameter combinations with different seeds.

Calculating all the possible hyperparameter combinations of the hyperparameters presented in Table 4.2 we see that 576 MLP networks are trained per sweep. We select the structural hyperparameters to find MLPs with sufficient representational capabilities while still being feasible to train with the available computational resources. The learning rate and batch size are selected over a wide range of values while still being memory

conscious.

Table 4.2: The value ranges which the initial hyperparameter grid search uses to search for optimal architectural and training hyperparameters.

Parameter	Values
Network width	10, 100, 1 000
Network depth	1, 2, 3, 4
Learning rate	1e-02, 1e-03, 1e-04, 1e-05
Batch size	16, 32, 64, 128

We select the best performing MLP based on validation loss from this hyperparameter sweep to evaluate the unseen test set.

4.5 Results

In this section, we use the datasets and training protocol discussed to investigate possible feature representation, the effect different hyperparameter choices have on performance, and the final observed performance of the MLP architecture.

4.5.1 Feature representation

We start this analysis by using the angle and absolute values of the CSI symbol produced by the pilot symbols to generate the features presented to the MLP as input and target. These features were used by Erdogmus *et al.* [67] to represent QI symbols for a similar channel equalisation problem.

Noiseless data

The first model is trained using this feature representation and the noiseless dataset. This combination is also used to validate the training process. We expect a BER of 0 from the

trained MLP, as an LS implementation produces CSI that can obtain a BER of 0 results on this dataset.

After the hyperparameter search has concluded, we find that the best hyperparameter combination produces an MLP with a loss of 4.61e-2 and a BER of 8.33e-4 on the test set. As the BER of the MLP is not 0, we suspect that the MLP has trouble training on the data. In analysing the MLP's estimations, we investigate both the network's absolute and angle feature outputs. We find the MLP does well predicting the absolute value of the Noiseless dataset as it is a fixed value for this dataset, as discussed in Section 3.4.1, due to not having a delay profile. However, the network has difficulty estimating the CSI's angle feature. We see the incorrectly estimated angles in Table 4.3 and note that these values lie close to the discontinuity found in the radian angle representation of the CSI. This discontinuity exists where the angle representation moves from 3.14 (π) to -3.14 ($-\pi$). This is where the CSI moves past 180 degrees from its original starting position. Instead of recognising the discontinuity, the MLP estimates a value between these two points.

Table 4.3: Predicted CSI angle of two OFDM signals compared to the target CSI angle over the same OFDM signals.

	Predicted CSI (OFDM symbol)		Target CSI (OFDM symbol)	
Subcarrier	1	2	1	2
1	**1.36**	**1.35**	3.09	3.09
2	3.05	3.05	3.05	3.05
3	3.00	3.01	3.01	3.01
4	2.88	2.89	2.96	2.96
5	2.85	2.85	2.92	2.92
6	2.80	2.8	2.88	2.88
7	2.77	2.77	2.83	2.83
8	2.72	2.72	2.79	2.79
9	2.67	2.68	2.75	2.75
10	2.56	2.55	2.71	2.71
11	2.52	2.51	2.66	2.66
12	2.46	2.45	2.62	2.62

As we suspect that batch normalisation is part of this problem (for reasons explained in Section 4.5.2) we investigate this further. As described in Section 4.5.2), we find that by removing batch normalisation we do obtain a BER of 0 on the noiseless dataset, using

the same feature representation as studied up to this point.

Noisy data

We continue experimenting with the same data representation and train MLPs on the Noisy dataset. While comparing the networks trained on the noiseless dataset and the network trained on the noisy dataset we find more concerning behaviour: Applying each of these networks to the Noisy dataset's test set, we find the noiseless MLP achieves a lower BER, of 5.16e-3, than the network trained on the Noisy dataset, with a BER of 1.72e-2. This is interesting behaviour as we expect networks trained on a specific data distribution to outperform networks trained on alternative distributions. Inspecting the loss values of these networks we find that the MLP trained on noisy data obtains a lower loss than the noiseless network, with a loss of 1.28e-1 and 2.31e-1, respectively.

Not only does the network trained on a different dataset outperform the noisy dataset trained MLP, but there is a mismatch between the loss and the BER of the MLPs. These results characterise an unreliable training process as we expect the loss and BER to be correlated: QI symbols of estimated CSI that are close to the real CSI in Euclidean space are expected to achieve low BERs.

Once again, when examining the estimated features, we find that adding noise has reintroduced the discontinuity problem in the angle feature. This can be expected as the introduction of noise has made the identification of the exact point of discontinuity more troublesome for the MLP. In investigating discontinuities in the input space of neural networks, we came across the work of Ahmed *et al.* [68]. Ahmed *et al.* [68] describe how neural networks perform better with data represented in a Euclidean space without discontinuities than in non-Euclidean spaces. For this reason, we change the extracted features from angle and absolute (amplitude) values to quadrature and in-phase amplitude, which are Euclidean features without discontinuities.

We retrain the MLP on the noisy dataset, now representing the CSI features in QI format, and find that the best performing hyperparameter combination obtains a loss of 5.23e-3

and a BER of 0 on the noisy test set.

Conclusion

Based on these results, we chose to continue working with Euclidean data as features for CSI estimation. We make this decision as QI features obtain better results than the angle and absolute features as it removes the discontinuity in the data representation, giving a reliable link between MMSE and BER. This compatibility makes the loss function a reliable indicator of BER. This quality was expected from CSI from the beginning of this study as CSI equalises received signals by division, meaning that an error in CSI would translate linearly to the equalised data.

4.5.2 Effect of hyperparameter choices

In training the MLP for CSI estimation we investigate the effect of structural hyperparameters, training hyperparameters, and batch normalisation on the datasets.

Structural hyperparameters

We examine the effect of structural hyperparameters on the MLP when training on the noiseless dataset and employing the angle and absolute representation by evaluating the validation set performance average over three seeds in Table 4.4. From this table we see that networks with wider and deeper layers provide the best performance of the trained networks, indicating that an increase in representational ability increases the MLP's ability to estimate CSI. We do not however further extend the hyperparameter search grid as an increase in performance is diminishing and the larger architecture poses more computational complexity to train.

Since we can show that larger networks provide better results, we only train MLP networks with an architecture of 4 layers and 1 000 nodes per layer for the delay profile and Doppler

Table 4.4: Best performing structural hyperparameters validation set MSE loss for MLP trained on noiseless data.

	Depth			
Width	1	2	3	4
10	1.05e-1	9.92e-2	9.29e-2	6.12e-2
100	5.45e-2	4.39e-2	4.39e-2	4.07e-2
1000	5.36e-2	3.73e-2	3.51e-2	3.29e-2

spread datasets. Furthermore, we do not extend the range hyperparameter sweep for the structural parameters as we only need an architecture with enough representational ability to provide consistent CSI estimations.

Training hyperparameters

Further investigating the initial noiseless dataset we find that a learning rate of 1e-3 provides the best performance for all batch sizes. We also observe that the performance increases as batch sizes increase, prompting us to increase the hyperparameter search range. We extend the batch size range to {32, 128, 256, 512}, allowing more data to be passed through the MLP before a gradient update is done. As the optimal learning rate is found in the middle of the search range, the learning rate search range remains unchanged.

Batch normalisation

In Section 4.5.1, we found that using non-Euclidean data for an MLP trained on noiseless data was troublesome. The failure of the network to compete with LS CSI estimation leads us to investigate architectural hyperparameters. Since layer width and depth do not affect the MLP's ability to identify the position of the discontinuity, we look to the activation function and batch normalisation layers. We have seen TanH used in the CSI estimation field before, so we chose to investigate batch normalisation.

It has been suggested by Santurkar *et al.* [69] that batch normalisation does not reduce

internal covariance shifts, as stated by S. Ioffe *et al.* [48], but instead smooths the optimisation landscape, leading to more predictable gradient behaviour. Santurkar *et al.* [69] state that this is possible as batch normalisation reparameterises the underlying optimisation problem. We hypothesise that smoothing of the optimisation landscape by batch normalisation is why the MLP has difficulty identifying the discontinuity. This hypothesis is formed since the region of the optimisation landscape where the discontinuity is recognised may be a very sharp point that is possibly being smoothed over.

We test this hypothesis by retraining the MLP on noiseless data using the angle and absolute representation but removing batch normalisation layers. Comparing the architecture hyperparameter results from the sweep in Table 4.4 to the results in Table 4.5, we find that larger networks do several orders of magnitude better when batch normalisation has been removed. Testing these new hyperparameter combinations on the noiseless test set, we obtain a BER of 0 as reported in Section 4.2.

Table 4.5: Best performing structural hyperparameters validation set MSE loss results for MLP trained on noiseless data and no batch normalisation layers.

	Depth			
Width	1	2	3	4
10	8.41e-2	9.45e-2	1.27e-1	1.95e-1
100	5.41e-2	4.46e-2	2.54e-2	1.14e-3
1 000	4.71e-2	1.65e-3	9.49e-8	6.33e-8

These results show that batch normalisation has a smoothing effect on the results between different hyperparameter combinations, providing more consistent results in the hyperparameter landscape. This is stated as more diverse results are found in the hyperparameter landscape when batch normalisation is removed. These more diverse results lead to MLPs that can recognise the position of discontinuity in the feature representation.

We further investigate the properties of batch normalisation in the hyperparameter landscape by training an MLP on the noisy dataset using the QI representation. By performing two hyperparameter sweeps, with and without batch normalisation, we investigate the effect of batch normalisation on Euclidean features. We report these results over the training hyperparameters in Table 4.6 and Table 4.7, including and not including batch

normalisation, respectively.

Table 4.6: Validation set MSE loss results over grouped training parameter for noisy data with QI values as features.

	Learning rate			
Batch size	1e-5	1e-4	1e-3	1e-2
32	5.28e-3	5.22e-3	5.49e-3	2.64e-2
128	5.32e-3	5.22e-3	5.49e-3	1.40e-2
256	5.33e-3	5.26e-3	5.56e-3	9.37e-3
512	5.36e-3	5.31e-3	5.56e-3	7.59e-3

Table 4.7: The same setup as in Table 4.6 but including batch normalisation.

	Learning rate			
Batch size	1e-5	1e-4	1e-3	1e-2
32	5.47e-3	5.64e-3	5.37e-3	6.33e-3
128	5.47e-3	5.50e-3	5.36e-3	5.75e-3
256	5.48e-3	5.48e-3	5.47e-3	5.66e-3
512	5.46e-3	5.47e-3	5.57e-3	5.59e-3

From the results shown in Table 4.6 and Table 4.7 we see that adding batch normalisation provides consistency between results in the hyperparameter landscape. However, removing batch normalisation provides us with the best performing network from both sweeps. This network's performance increase is negligible, so we choose to include batch normalisation in future networks. We make this choice as a smoother hyperparameter landscape reduces training time, leading to fewer search grid extensions being needed in the optimisation process.

Conclusion

It is thus concluded that while batch normalisation has a desirable effect on the training process, it may be detrimental to certain feature representations. We can, however, mitigate the harmful effects of batch normalisation by using Euclidean data representations and possibly including networks without batch normalisation when probing new datasets.

4.5.3 Observed performance

We test the ability of the training protocol and MLP architecture by training MLPs on the three delay profile datasets. This training protocol is implemented as described in Section 4.4 with the difference that all networks are 4 layers deep and 1 000 neurons wide, along with the adaptation made to the batch size search range in Section 4.5.2.

We examine the performance of the best networks on both their test set and the test sets of the other delay profiles in Table 4.8. The BERs show that networks perform best on the delay profile they were trained on and are comparable to or outperform results obtained by an LS implementation. This degree of cross-condition generalisation, especially with neighbouring delay profiles, is promising, as delay profiles are continuous and ever-changing in real-world channels.

Table 4.8: BER of MLP networks trained on one delay profile train set and applied to the same or other delay profile test sets.

Trained on	EPA	EVA	ETU
EPA	3.97e-3	1.10e-2	3.63e-2
EVA	4.82e-3	6.88e-3	1.57e-2
ETU	6.69e-3	8.35e-3	1.16e-2
LS Method	1.20e-2	1.43e-2	3.28e-2

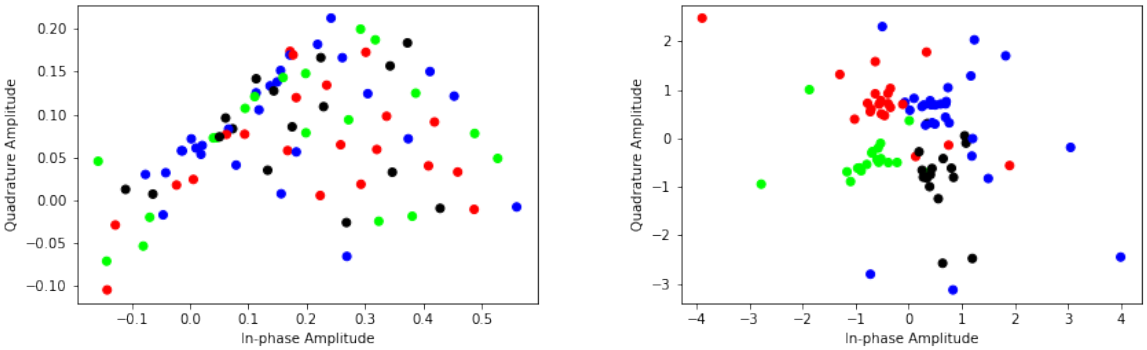
We further test the ability of the MLPs by training on the Doppler dataset. In Table 4.9, we observe that the best performing hyperparameter combination outperforms LS and is comparable with MMSE at lower Doppler frequencies.

Table 4.9: BER of different equalisation methods on the Doppler dataset with various amounts of Doppler spread.

Doppler spread (Max Hz)	MLP	LS	MMSE
50	6.99e-3	1.38e-2	8.6e-3
100	8.01e-3	1.45e-2	8.6e-3
200	9.13e-3	1.5e-2	8.6e-3

Investigating the cases in which bit errors are made for the Doppler dataset, we find that most errors are made when the CSI estimates approach zero. In Figure 4.2, we observe

both the QI of the CSI and equalised bits produced by the MLP. CSI points estimated around zero lead to exploding symbols in the equalised symbols. This is due to the noise enhancement problem discussed in Section 3.7 and is, for the time being, unavoidable.



(a) CSI estimation

(b) Equalised symbols

Figure 4.2: CSI estimate produced by the MLP in attenuated channel conditions and the resulting equalised symbols obtained when equalising with the estimated CSI.

4.6 Conclusion

This chapter uses a sound training protocol to analyse the effect of different data representations and hyperparameter choices. In the process, we found that data representations containing discontinuities are unsuitable for our use case. Furthermore, we find how batch normalisation can be removed to help find discontinuities in the feature representation but is better utilised to reduce variation in hyperparameter sweep results. Finally, using appropriate hyperparameters and feature representations, we train MLPs capable of contending with MMSE CSI estimators in Doppler channel conditions.

Moving forward, we advance the architecture to convolution layers that perform better on data where neighbouring points have location-based correlations, as is the case in our data. Furthermore, we look to improve generalisation by training networks over an SNR range instead of only a single SNR. By attending to these aspects, we move closer to understanding deep learning for CSI.

Chapter 5

CSI estimation using ResNet

In this chapter we compare the generalisation ability of a CNN-based architecture to an MLP architecture using different noise distributions in datasets.

5.1 Introduction

This chapter investigates using a ResNet model on the CSI estimation task and developing an appropriate dataset structure for the rest of this study. First, we discuss the ResNet architecture and how the dataset's samples are adjusted for the CNN layers. The ResNet architecture is then trained and tested to evaluate its generalisation ability on different SNR. Finally, a combined SNR dataset is proposed, developed and tested. Note that sections of this chapter has been published by Oosthuizen *et al.* [21].

5.2 ResNet architecture

In Chapter 4, we use an MLP architecture to estimate CSI. The drawback of this architecture is that the two-dimensional slot is reduced to a one-dimensional tensor. By losing one dimension, we lose valuable spatial-temporal information. CNNs, however, can take the correlation between neighbouring subcarriers into account in several dimensions with their convolution layers. This feature of CNNs is important as Delay profiles and Doppler spread introduce fluctuations of CSI in both the time and frequency domains.

For the architecture in this chapter, we use the adaptation of the generator network proposed by Zhao *et al.* [16]. This generator network is a residual network containing four residual blocks, as discussed in Section 2.3.1, and relies on pre-upscaling of CSI data from pilot symbols, as opposed to the built-in upscaling of the original SRGAN, as can be observed in Figure 5.1.

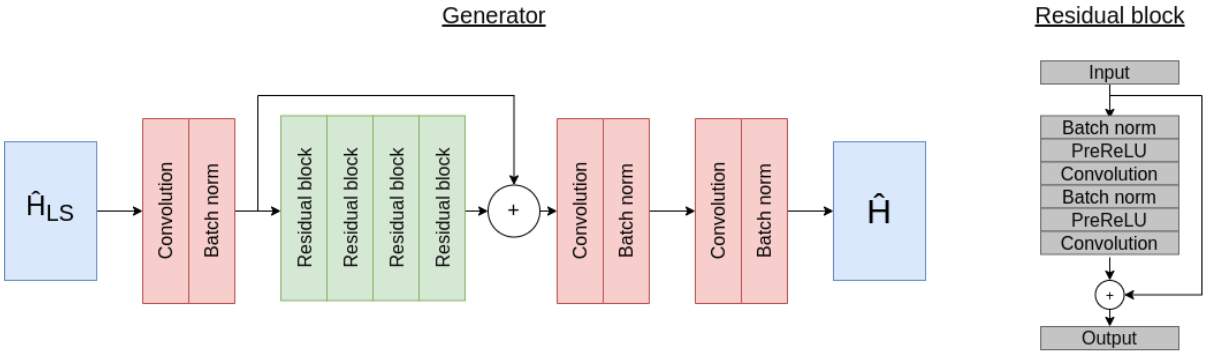


Figure 5.1: Illustration of the modified SRGAN's generator network developed by Zhao *et al.* [16] and used in this study.

5.3 Data samples

In order to use the ResNet architecture we need to present our data samples in the appropriate manner. We use slots with 4-QAM modulated symbols for the samples. After the CSI has been extracted using the 4 pilot symbols, the LS algorithm is used as a

pre-upsampling method. An LS estimate of the same dimensions as the target CSI is thus used as input samples for the ResNet. To better replicate the input data of SRGAN (a three-feature RGB image), we extend the features to the absolute value of the QI symbols, the normalised Quadrature (Q) amplitude, and the normalised In-phase (I) amplitude. We normalise the QI components by dividing the amplitudes with the absolute value of the QI symbols. Thus, we fully take advantage of the spacial temporal nature of the data with the CNN by supplying the input as a 12x7x3 data sample, as can be seen in Figure 5.2.

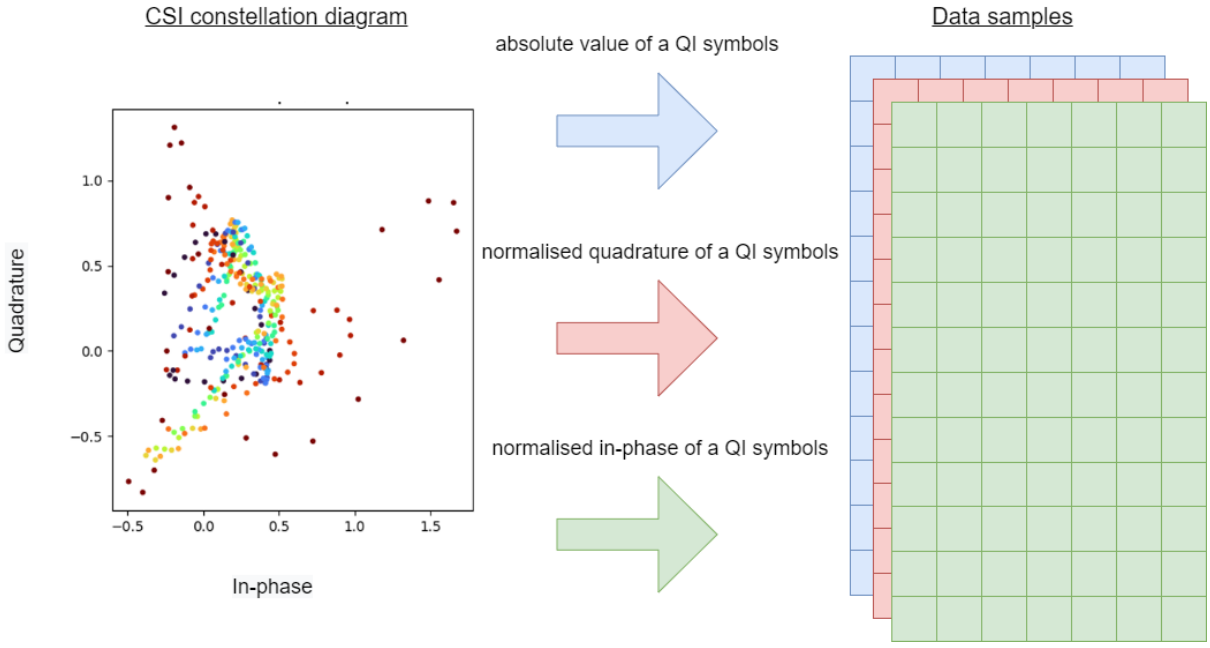


Figure 5.2: An illustration of a CSI constellation diagram with 10 dB antenna noise being translated to an input data sample with three features. Each QI symbol translates to a 1x3 point in the data sample and the data sample dimensions correspond to the subframe size selected, in this case, a slot.

5.4 Noise generalisation ability

In this section, we test the ability of the ResNet to generate CSI by training networks using a hyperparameter search. We repeat this for three SNR datasets. By establishing a training protocol to train networks with, we are able to evaluate the generalisation

ability of the ResNet. We compare the generalisation ability of the ResNet to the MLP architecture used in Chapter 4.

5.4.1 Training protocol

We train all models using the Adam optimiser and MSE loss function. In addition, we look only to optimise learning parameters for the hyperparameter search since the architecture has shown good performance in similar CSI estimation environments [16]. As indicated in Table 5.1 we sweep over batch size, learning rate and weight decay. All sweeps are done over three initialisation seeds and the result reported is the average of three networks.

Table 5.1: The value ranges that the initial hyperparameter grid search uses to search for optimal training hyperparameters.

Parameter	Values
Learning rate	1e-03, 5e-04, 1e-04, 5e-05, 1e-05
Batch size	4, 16, 32, 64, 128
Weight decay	0.1, 0.5, 0.9

The dataset for this experiment is generated for channels with 1 dB, 10 dB and 20 dB noise, resulting in three datasets. Datasets contain 20 000 training samples, 5 000 validation samples and 5 000 test samples, the same as in the previous chapter. Noise levels of 1 dB, 10 dB and 20 dB are selected as SNR as they are spaced sufficiently and above the noise floor at 0 dB and in the range where BER would not be 0 for MMSE estimators. All datasets used to train the network in this chapter are simulated using a channel with an EVA delay profile and 50 Hz of Doppler spread.

5.4.2 Generalisation results

The hyperparameter search results indicate that the best performance is obtained at a small batch size and learning rate, while a weight decay value of 0.5 performs best. The BER of the best performing hyperparameter combinations on the test set of their trained

SNR and neighbouring SNR are depicted in Table 5.2. We see that each network performs best on the dataset they are trained on can generalise over different SNRs.

Table 5.2: BER of ResNet networks trained on different SNR and tested over all datasets' test sets.

Trained on SNR (dB)	Tested on 1 dB	Tested on 10 dB	Tested on 20 dB
1	1.95e-1	5.00e-2	1.53e-2
10	1.99e-1	4.43e-2	7.35e-3
20	2.14e-1	4.64e-2	6.78e-3

The ResNets trained on 10 dB SNR show the smallest drop in generalisation performance between SNR compared to the other training datasets. This is because 1 dB SNR is extremely noisy, and the data distribution is not similar to the distribution of 20 dB data and vice versa. The data distribution of the 10 dB dataset is in between these two extremes and thus provides the best generalisation of the three datasets.

5.4.3 Generalisation comparison

We compare the results of ResNet's generalisation ability to the final MLP architecture used in Chapter 4 by training the MLPs on the same datasets used to test ResNet in this chapter. From the BER results in Table 5.3 we can see that the ResNet outperforms the MLP results in performance and generalisation in all cases except one. This improvement in BER performance between architectures supports our decision to focus only on convolution-based architectures.

Table 5.3: BER of MLP networks trained on different SNR levels and tested over all datasets' test sets.

Trained on SNR (dB)	Tested on 1 dB	Tested on 10 dB	Tested on 20 dB
1	2.01e-1	5.49e-2	2.64e-2
10	2.02e-1	4.76e-2	9.34e-3
20	2.08e-1	4.94e-2	6.98e-3

5.5 Combined SNR dataset

Knowing that both CNN and MLP networks have reduced performance due to weak generalisation over the SNR range, we investigate an alternative training strategy to help improve generalisation. In this section, we propose and investigate the effects of a set of combined datasets containing a mixture of all the SNRs using a relevant metric and decide on which distribution performs best for the ResNet architecture.

5.5.1 Experimental protocol

For this experiment, we create weighted datasets of 30 000 samples: 20 000 training samples and 10 000 validation samples. Since we know that 10 dB SNR provides the best generalisation, we investigate which SNR, low or high, gives the best generalisation performance if the number of samples is unevenly distributed to one side. We thus inspect three training datasets: evenly distributed, high noise distributed, and low noise distributed. The distribution of samples in each dataset is described in Table 5.4.

Table 5.4: The number of SNR samples included in each distributed dataset.

Dataset	Number of 1 dB samples	Number of 10 dB samples	Number of 20 dB samples
Even Noise Distribution Dataset (ENDD)	10 000	10 000	10 000
High Noise Distribution Dataset (HNDD)	15 000	10 000	5 000
Low Noise Distribution Dataset (LNDD)	5 000	10 000	15 000

We retrain the models using a similar hyperparameter search, with weight decay kept at 0.5, and the learning rate refined to {5e-04, 1e-04, 1e-05}.

5.5.2 Distributed dataset results

The hyperparameter search on the new datasets shows that the networks still perform best when trained using small learning rates and batch sizes. We observe consistent results between datasets by evaluating the best hyperparameter combinations' generalisation ability and performance in Table 5.5. However, we see that the HNDD provides better BER performance for 1 dB SNR and the same for the LNDD for 10 dB and 20 dB SNR. Furthermore, all combined datasets perform close to the individually trained networks in Table 5.3, thus showing generalisation over the SNR range with no additional training or architecture costs. The LNDD improves on the individually trained BER results for 10 dB and 20 dB SNR data due to the inclusion of noisier data in the training set.

Table 5.5: BER of MLP networks trained on different SNRs and tested over all datasets' test sets.

Trained on dataset	Tested on 1 dB	Tested on 10 dB	Tested on 20 dB
ENDD	1.95e-1	4.45e-2	7.02e-3
HNDD	1.95e-1	4.46e-2	7.56e-3
LNDD	1.97e-1	4.44e-2	6.67e-3

5.5.3 PICDB

An extension of BER used in this study is PICDB, a comparative metric showing the percentage increase of bits correctly demodulated between two BERs. We use this metric as some BERs in this study are too close to observe the impact on figures or tables accurately. PICDB also has a normalising effect on different SNRs as we make use of a percentage, enabling different SNRs to be compared. We calculate PICDB by dividing the ratio of correctly classified bits of our BER_1 by the ratio of correctly classified bits of our reference BER_2 before calculating the percentage in Equation 5.1. From this equation, we gather that if the PICDB provides a positive value, it means that BER_1 performs better than the reference BER_2, while if PICDB is negative, it means that BER_1 has less correctly classified bits than BER_2.

$$PICDB = (\frac{1 - \text{BER}_1}{1 - \text{BER}_2} - 1) * 100 \tag{5.1}$$

5.5.4 Final results

In selecting a combined dataset for further training, we ensure that the dataset that provides the best performance in correctly decoded bits is selected. For this reason, we introduce PICDB, which uses the BER metric to show the difference in classified bits between two networks.

By examining the PICDB of the single SNR trained network against the distributed datasets in Figure 5.3 and the generalisation results in Table 5.5 we decide to use the ENDD. In this figure, we see the large percentage of incorrectly classified bits the LNDD has for 1 dB SNR, while the HNDD performs the same as the ENDD. Examining the area under each graph makes the ENDD approach the clear choice for further datasets.

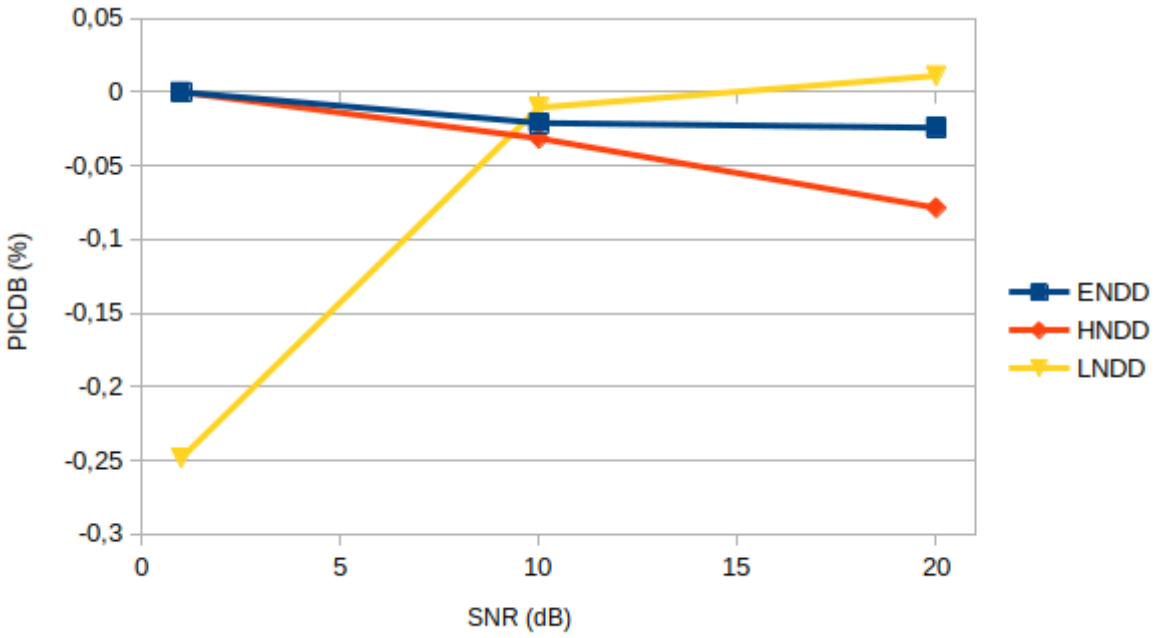


Figure 5.3: PICDB of the CSI generated by estimators trained on the ENDD, HNDD and LNDD with respect to single SNR trained CSI estimators, shown over the test set.

Evaluating the performance of the networks trained with a single SNR (separate networks), the ENDD, LS estimation and MMSE estimation in Figure 5.4, we see that the ENDD is comparable with the separately trained networks. Furthermore, we observe that we outperform MMSE and LS estimation methods with this CNN implementation using only slot-sized samples. This dataset thus allows the network to optimise its approach to the multi-SNR problem in the training phase and improve generalisation results over the entire range.

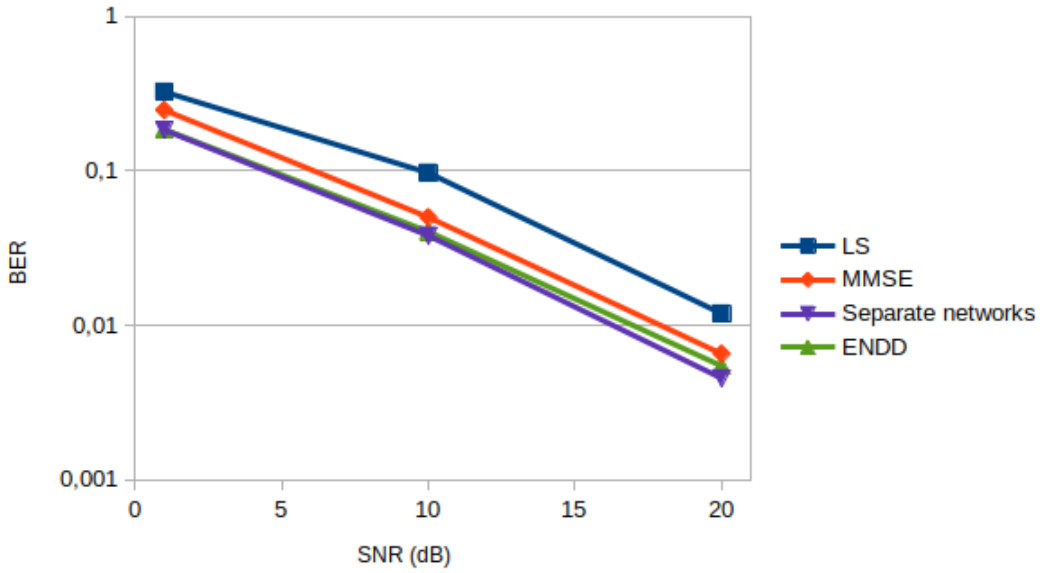


Figure 5.4: BER of the combined dataset training method, single SNR trained networks, MMSE and LS estimators over the SNR range, shown over the test set.

5.6 Conclusion

This section introduces a ResNet architecture using LS estimations as inputs to the CSI estimation task. The ResNet, which is capable of extracting spatial-temporal features, outperforms the MLP architecture from Section 4 when trained on a dataset consisting of samples with single SNR of noise. We thus move forward using ResNet which uses a multidimensional input compared to the MLP's input that removes spatial-temporal coherence.

We find that ResNet networks generalise poorly when trained only on a single SNR of noise, so we investigate distributed SNR datasets, which are datasets containing samples with different amounts of SNR. Training networks using these datasets, we obtained networks comparable with the single SNR trained networks. In fact, the network trained on a distributed SNR dataset generalises better than any of the networks trained on a single SNR dataset; we thus have single networks able to contend with three networks over the SNR range due to an improved training set. Evaluating the final BER performance of the networks trained in this section and baseline methods, we find that the ResNet networks outperformed the LS and MMSE estimators.

Moving forward, we further refine the datasets by adding more SNR increments to the range of noise represented in the dataset. We also consider increasing the datasets' sample sizes and introducing adversarial training to further increase network performance.

Chapter 6

CSI estimation using GANs

This chapter introduces adversarial training to the CSI estimation task in a SISO environment.

6.1 Introduction

In this chapter, we investigate the effect of adversarial training on CSI estimation, as well as how different input sample sizes affect network performance. We first introduce the dataset used in this section and describe how the training protocol is adjusted for adversarial training. Next, we discuss the three adversarial architectures used in this study and evaluate initial results. Finally, we examine the effect of increased input sample sizes on these adversarially trained networks. Note that sections of this chapter has been published by Oosthuizen *et al.* [21].

6.2 Dataset

After the exploratory experiments into the simulation and dataset in Chapters 4 and 5, we present the Enhanced Even Noise Distributed Dataset (EENDD). After discussing the EENDD we compare the results of ResNet networks trained on previous datasets to the EENDD to ensure that additions to the dataset are not hindering network performance.

6.2.1 Enhanced even noise distributed dataset

In this section, we use the evenly distributed SNR method tested in Section 5.5 to generate the EENDD on which to train and test the networks. We expand the SNR used in the dataset from 1 dB, 10 dB and 20 dB SNR to SNR incremented in steps of 1 dB from 1 dB to 20 dB, with all SNRs having the same number of samples in the dataset. Furthermore, we generate three datasets for this chapter, each with a different input sample size: 12x7, 24x14 and 120x14. The total size of each dataset is increased to 40 000 training samples, 10 000 validation samples and 10 000 test samples. We continue using an EVA delay profile with 50 Hz Doppler spread as channel impairments. Finally, we use 16-QAM modulated data for evaluation instead of 4-QAM, as 4-QAM only requires the CSI to correct the symbol phase, while 16-QAM requires accurate amplitude correction as well.

6.2.2 Dataset evaluation

To evaluate the validity of the EENDD, we compare the results of networks trained on the EENDD and the distributed SNR datasets introduced in Section 5.5, using the training protocol described in Section 6.3. We observe the PICDB of these networks using the EENDD's unseen test set to compare ResNets trained on the EENDD to ResNets trained on the distributed SNR datasets in Figure 6.1.

From Figure 6.1, we observe that the HNDD performs the worst on almost all SNRs while the ENDD outperforms the other datasets after 9 dB SNR. Unlike the results found in

Section 5.5, the HNDD and LNDD do not perform best on the SNRs they were trained on. The performance shift towards the ENDD can be attributed to the inclusion of amplitude in the demodulation process, as 16-QAM modulation differentiates between symbols in both amplitude and phase, which was not the case with the 4-QAM symbols studied in Chapter 5.

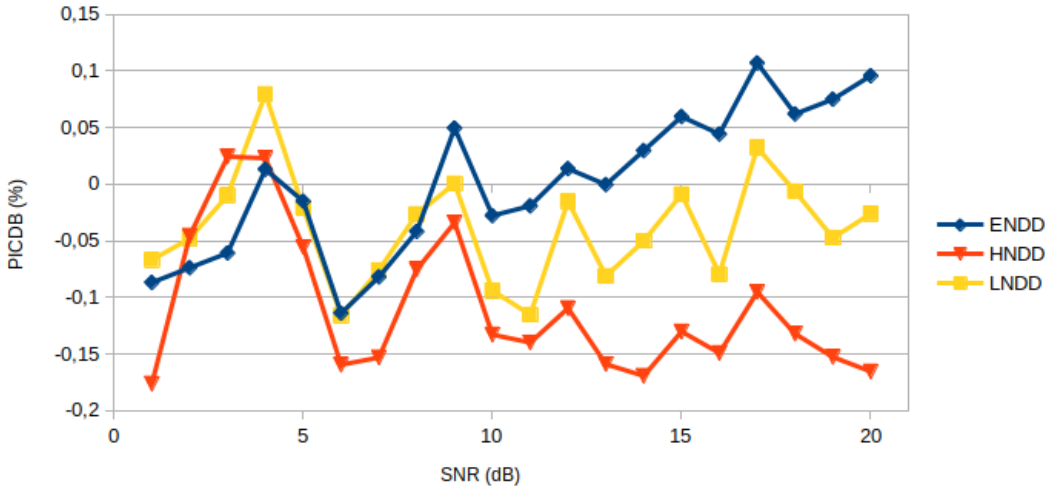


Figure 6.1: PICDB of the data demodulated by CSI generated from networks trained on SNR distributed datasets with respect to EENDD CSI estimators, shown over the test set with 16-QAM symbols using 12x7 input sample sizes.

Averaging the PICDB result over the SNR range for the distributed SNR training sets against the EENDD, Table 6.1, shows that the ENDD training set provides the best performance. However, the margin by which the ENDD outperforms the EENDD is so minor that we consider this as equal performance.

Table 6.1: Average PICDB over all SNRs of ResNet networks trained on different noise distributed datasets compared to the EENDD.

Data distribution	ENDD	HNDD	LNDD
PICDB (%)	1.45e-3	-1.09e-1	-3.88e-2

The performance of the EENDD against LNDD and HNDD again shows that the correct distribution has been selected. The minor margin between the ENDD and the EENDD indicates that adding samples for more SNR's values to the dataset does not affect the

overall performance. Furthermore, indicating that enough training samples were included in the ENDD and this applies as well to the EENDD.

6.3 Training protocol

In this chapter, we train four architectures over three sample sizes equalling 12 different network architectures. The four architectures being compared is the standard ResNet followed by ResNet networks trained using adversarial training methods. Each network architecture will go through an independent hyperparameter sweep to ensure that a good architectural and training hyperparameter combination is found for the specific conditions.

We once again train the networks using Adam as an optimiser. The initial hyperparameter search is conducted as shown in Table 6.2. Initial testing also included adding different amounts of noise to samples before presenting them to the discriminator and not training the discriminator every epoch. These methods, however, did not show significant changes in generator performance and were left out of the hyperparameter search. Hyperparameter sweeps are extended from this initial sweep if the best performing results are found on the edge of the search grid. The full hyperparameter search range for each network and dataset combination can be found in Appendix A.2. Due to the computational expense of adversarial training, we do sweeps over one randomisation seed and retrain the best hyperparameter combination over two additional seeds to ensure consistent and repeatable results.

Table 6.2: Hyperparameter ranges used for the initial sweeps before being expanded to wider sweeps.

Hyperparameter	Value
Generator LR	1e-3, 1e-4, 1e-5
Discriminator LR	1e-3, 1e-4, 1e-5
Discriminator ratio	1e-2, 1e-3, 1e-4

6.4 Architectures

This section introduces the three adversarial training approaches and their evaluation using the EENDD. The generators and discriminators in this section each have identical architectures but differ in how they interact in the training process with the proposed training architectures. Discriminators are trained in the same manner as described in Section 2.5. Equations in this section are shown for single samples but are implemented in sample batches using an optimiser.

6.4.1 Adversarial ResNet

The first adversarial architecture we implement is the architecture proposed by Zhao *et al.* [16]. This architecture evaluates the generator G, the ResNet architecture introduced in Section 5.2, using the MSE between the output and the target sample CSI^{HR}, as seen in Equation 6.1:

$$l_{content} = MSE(G(CSI^{LR}), CSI^{HR}). \tag{6.1}$$

The discriminator D accepts the real sample as input and attempts to correctly classify the sample as real or fake using a BCE loss function, as observed in Equation 6.2:

$$l_{adversarial} = BCE(G(CSI^{LR}), 1). \tag{6.2}$$

The flow of the adversarial GAN's adversarial training is shown in Figure 6.2.

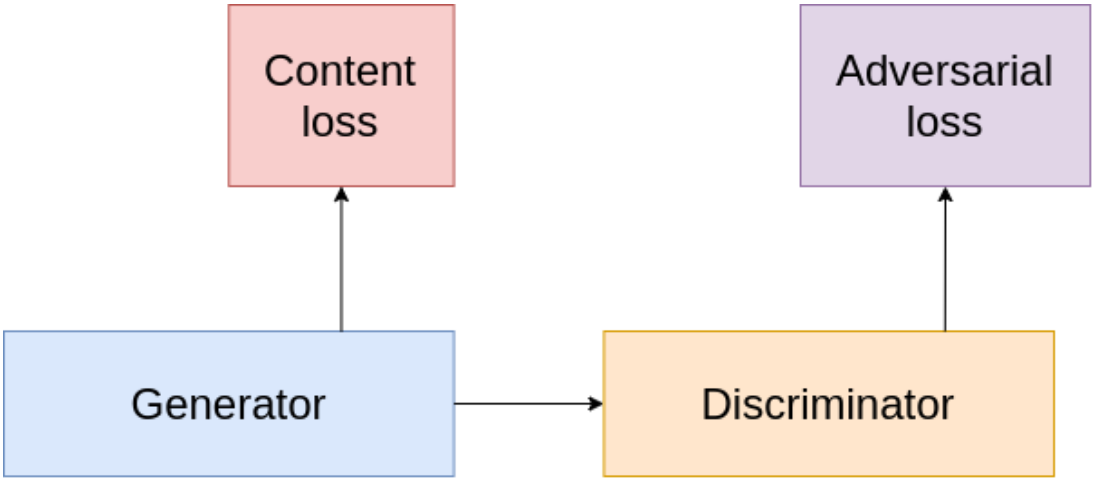


Figure 6.2: A representation of the composition of the content and adversarial loss in Adversarial ResNet.

6.4.2 SRGAN

We implement SRGAN as described by Ledig *et al.* [5] with the exception of the internal upsampling layers. The internal upsampling layers are removed so that all of the architectures have identical generator and discriminator architectures, ensuring that the method of adversarial training is analysed and not the architecture. SRGAN extracts the features ϕ of the CSI estimate provided by the generator and uses these features to calculate the content loss, as shown in Equation 6.3:

$$l_{content} = MSE(\phi(G(CSI^{LR})), \phi(CSI^{HR})). \tag{6.3}$$

The adversarial loss is obtained by providing the CSI estimate, before feature extraction, to the discriminator, as previously shown in Equation 6.2.

The flow of the SRGAN's adversarial training is shown in Figure 6.3.

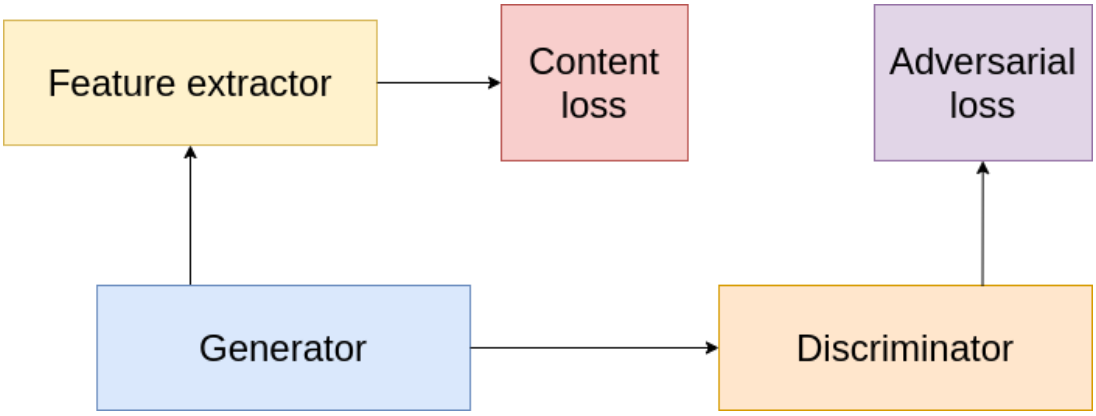


Figure 6.3: A representation of the composition of the content and adversarial loss in SRGAN.

6.4.3 Modified SRGAN

Modified Super Resolution Generative Adversarial Network (MSRGAN) is proposed in this study as a combination of adversarial ResNet by Zhao *et al.* [16] and SRGAN by Ledig *et al.* [5] The proposed network combines the CSI-based content loss with the feature extractors used in SRGAN. As a result, the content loss is obtained the same way as in Adversarial ResNet, using the MSE between the output and the target sample, as seen in Equation 6.4:

$$l_{content} = MSE(G(CSI^{LR}), CSI^{HR}). \tag{6.4}$$

MSRGAN, however, places a feature extractor between the generator and discriminator, thus providing the discriminator with feature maps instead of CSI estimates, shown in Equation 6.5:

$$l_{adversarial} = BCE(\phi(G(CSI^{LR})), 1). \tag{6.5}$$

The flow of the MSRGAN's adversarial training is shown in Figure 6.4.

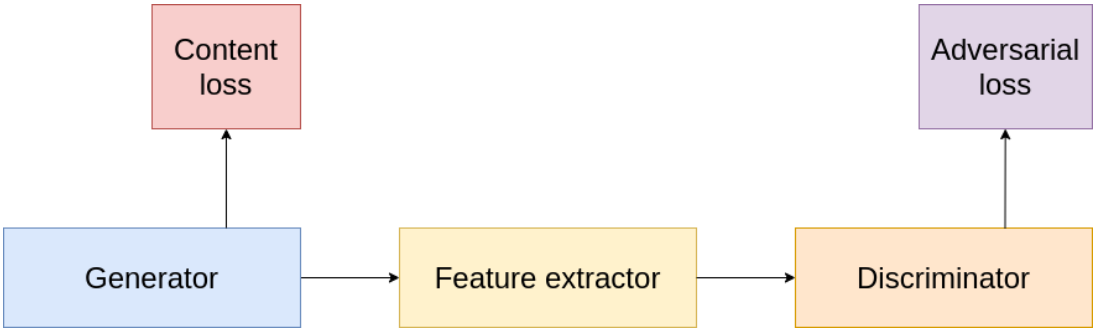


Figure 6.4: A representation of the composition of the content and adversarial loss in MSRGAN.

By providing the discriminator with extracted features instead of the generated CSI, we aim for accurate CSI estimations from the generator while obtaining feedback from the discriminator to increase the feature accuracy of the generator.

6.5 Adversarial training on small input size

This section evaluates the three proposed adversarial training methods against a non-adversarially trained ResNet, using a 12x7 input sample size.

We compare the adversarial ResNet, SRGAN and MSRGAN to non-adversarial ResNet using PICDB in Figure 6.5. The results in Figure 6.5 indicate that the two networks with feature extractors are outperformed on every SNR by adversarial and non-adversarial ResNet.

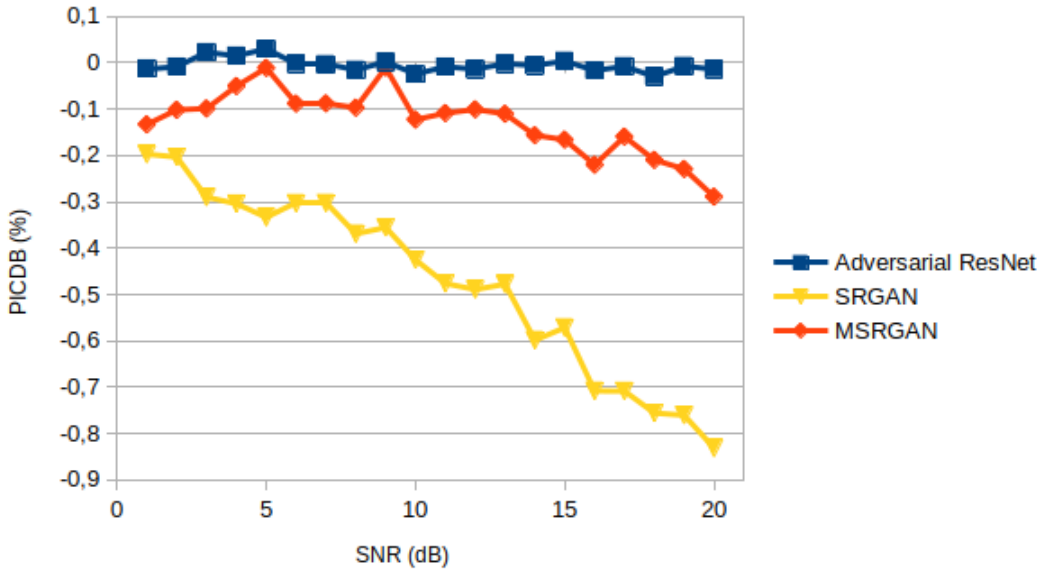


Figure 6.5: PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 12x7, shown over the test set.

Figure 6.5 illustrates that the feature extractor does not provide benefit in adversarial training for networks with a 12x7 input sample size. Furthermore, we observe that a content loss based on features, like in SRGAN, impairs the ability of the generator to make CSI estimations that are comparable with CSI-based content loss. Lastly, we observe that adversarial ResNet is comparable with non-adversarial ResNet over the entire SNR range when slot-sized input samples are used.

6.6 Adversarial training over increased input sizes

In this section, we evaluate the effect of increased input sample sizes on the adversarial and non-adversarial networks. As ResNet is the baseline, we first compare input sample sizes using the non-adversarial network before comparing adversarial to non-adversarial training over larger input sample sizes.

6.6.1 ResNet using different input sample sizes

To evaluate the impact of different input sample sizes on the networks' CSI estimation accuracy, we train three networks, each using a dataset with different input sample sizes. The selected sizes for the samples are 12x7 (as used thus far in the study), 24x14 (2 adjacent subframes) and 120x14 (10 adjacent subframes). We compare these networks using the unseen test sets against an LS estimation over the entire resource grid using PICDB in Figure 6.6. From these results, we see that moving from a 12x7 to 24x14 input sample size increases performance more than moving from a 24x14 to a 120x14 input sample size. We further note that the ResNet network obtains the most significant increase in performance at lower SNRs.

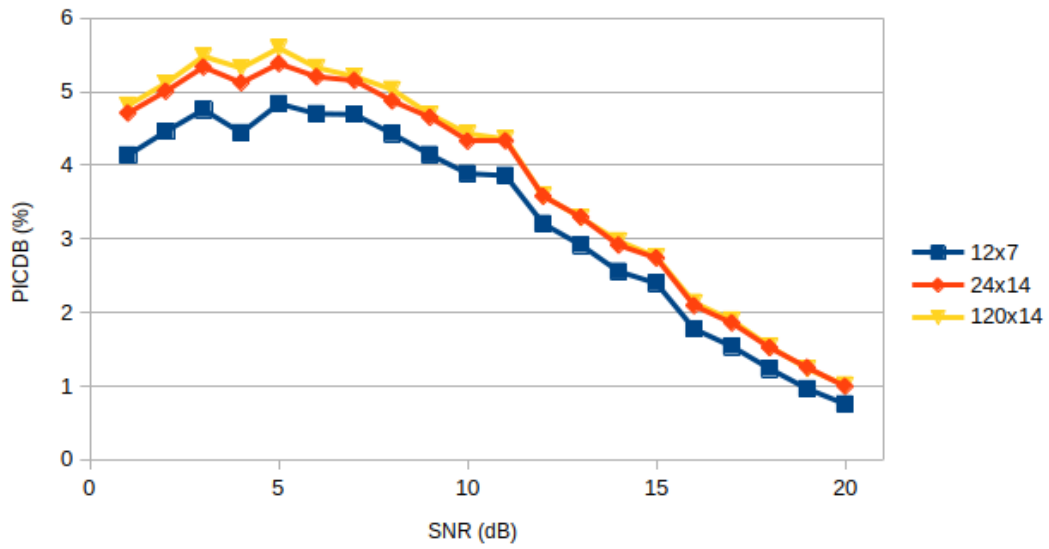
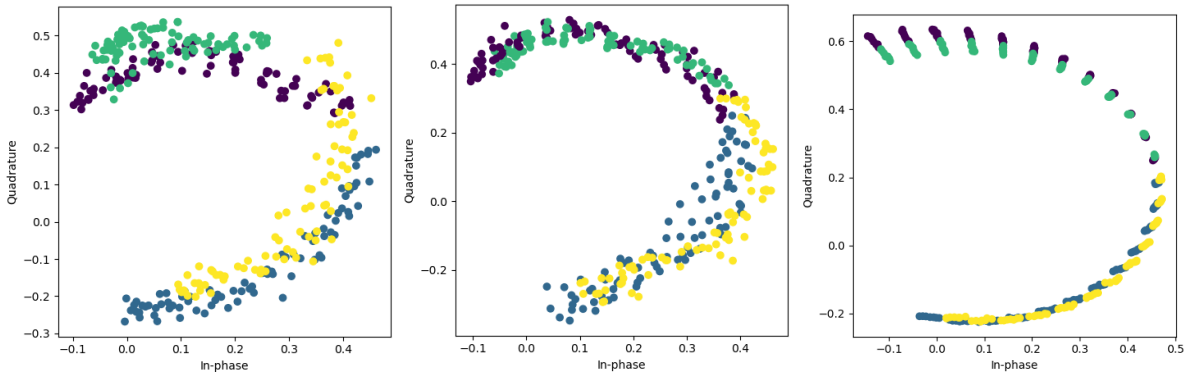


Figure 6.6: PICDB of the CSI generated by non-adversarially trained ResNet with respect to an LS estimation for three input sample sizes, shown over the test set.

When investigating the difference in CSI estimates between a 12x7 estimator and 24x14 estimator we can deduce why the increase in performance exists. We compare the estimators under the same CSI conditions by taking four 12x7 estimations and comparing the results to a single 24x14 estimation and the target CSI in Figure 6.7. Each colour in the figure represents a slot of data. Figure 6.7 thus compares the same number of subcarrier estimations, but with different input sample size networks. Comparing the two estima-

tors, we see that the 24x14 estimator obtains CSI slots which transition more seamlessly from one to the other in both the time and frequency domain. These smoother transitions are possible because the 24x14 estimator has more information on adjacent slots than the 12x7 estimator and thus leads to more accurate CSI estimations.



(a) Combined CSI of four 12x7 input estimations.

(b) CSI of a single 24x14 input estimations.

(c) CSI target for both 12x7 and 24x14 estimations.

Figure 6.7: Figures depicting the CSI estimates of four 12x7 estimators, one 24x14 estimator and the perfect CSI for the same channel conditions.

6.6.2 Adversarial training using different input sample sizes

Seeing the effect that increased input sample sizes have on ResNet performance over all SNRs, we train adversarial networks on these increased input sample sizes. We train the three adversarial networks on datasets with input sample sizes of 24x14 and 120x14 and compare these networks to a ResNet trained on the same input sample size.

Evaluating the comparison of the 24x14 input sample size in Figure 6.8, we observe that the networks perform slightly differently than when trained on 12x7 input sample slots in Figure 6.5. First, we can see that MSRGAN has increased its relative performance over the entire SNR, making it comparable with ResNet and Adversarial ResNet. However, SRGAN still underperforms with respect to other networks over all SNRs. These results indicate that the feature extractor used in MSRGAN is not hindering performance, as might have been indicated in Figure 6.5, but only requires larger input sample sizes to be

effective.

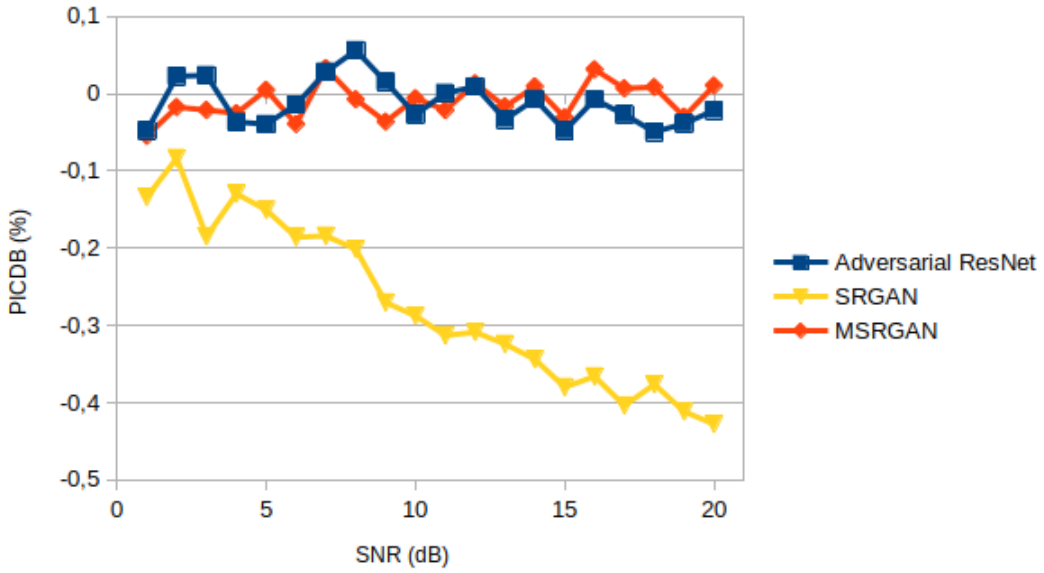


Figure 6.8: PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 24x14, shown over the test set.

Similar network performance is observed in the networks trained on 120x14 input sample sizes in Figure 6.9. Adversarial ResNet and MSRGAN are both still comparable with ResNet, while SRGAN is underperforming in all networks.

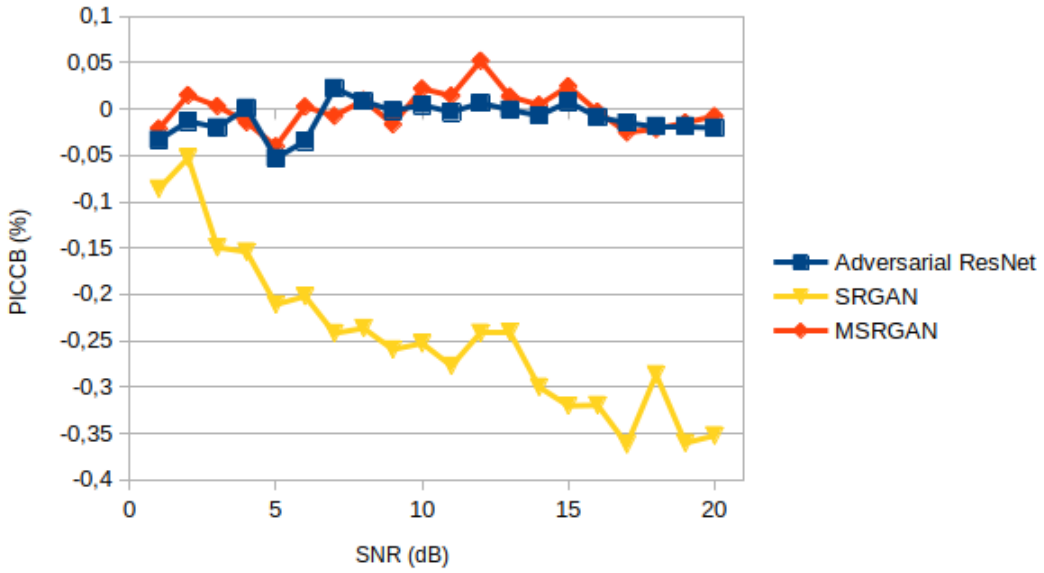


Figure 6.9: PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 120x14, shown over the test set.

By averaging the PICDB of the networks over the SNR range for each input sample size in Table 6.3, we can observe the overall result of each network. The results in Table 6.3 show that Adversarial ResNet only slightly underperforms ResNet over all input sample sizes, while SRGAN underperforms ResNet significantly. Finally, we observe that MSRGAN increases in performance as the input sample sizes increase. The correlation between input sample size and MSRGAN's performance is linked to the feature extractor network as adversarial ResNet does not follow this trend, and the feature extractor is the only difference in architecture between the networks.

Table 6.3: Average PICDB over all SNRs of adversarially trained networks trained on different input sample sizes compared to a ResNet of the same input sample size.

	Adversarial ResNet	SRGAN	MSRGAN
12x7	-4.97e-3	-4.73e-1	-1.27e-1
24x14	-1.20e-2	-2.73e-1	-9.36e-3
120x14	-9.75e-3	-2.45e-1	-4.27e-4

6.7 Conclusion

In this chapter, we use a finalised dataset to train adversarial architectures with different input sample sizes in a SISO environment. From the result, we deduce that the networks used in this chapter obtain increased performance as the input sample size increases. Furthermore, we also find that adversarial ResNet and MSRGAN contend with non-adversarial ResNet on larger input sample sizes. Finally, SRGAN consistently performs worse than the other networks used in this chapter.

Using these trained networks we move on to investigate their performance in MA environments.

Chapter 7

Analysis of multi antenna environments

In this chapter, we investigate how adversarial and non-adversarially trained networks, trained in a SISO environment, perform in MA environments and compare this to how the networks perform against the LTE benchmark.

7.1 Introduction

In this chapter, we apply the networks trained in Chapter 6 to MA environments to observe the effect of adversarial training in environments in which the networks were not trained. The networks are deployed in these MA environments to evaluate the generalisation of networks between environments. We first discuss the simulation parameters in which we will test these networks, followed by an analysis of the receiver diversity method. Finally, we discuss the results obtained from the MA environment and evaluate the results against LTE benchmarks. Note that sections of this chapter has been published by Oosthuizen *et al.* [21].

7.2 Simulation setup

To test how the trained networks perform in MA environments, we utilise the simulation developed in Chapter 3 to simulate MA environments. Using the simulation, we implement transmitter and receiver diversity as described in Section 3.6. We simulate the SNR range of 20 dB incremented in steps of 1 dB from 1 dB to 20 dB used to train the networks with an EVA delay profile and 50 Hz Doppler spread. Each SNR in the simulation is implemented over 20 resource grids with unique randomisation seeds for data and channel conditions that can be reproduced for each network. This simulation setup ensures that the results in this chapter are delivered at acceptable confidence levels for both BER and model variation as discussed in Appendix A.3.

From Figure 7.1, we see that we use already trained CSI estimation networks, that we train in the Python environment, in our Matlab® simulation. Due to the limitations imposed by Matlab®'s deep learning toolbox, we convert our PyTorch networks to Open Neural Network Exchange (ONNX) formatted networks. To achieve this, we use the ONNX package available in Python, which converts the trained generators from Chapter 6 to networks that can be imported to Matlab®. To ensure that the conversion is successful, we run inference on both formats using randomly generated inputs and obtain the same results. These converted models are then imported to Matlab® where they are implemented as CSI estimators, replacing our baseline LTE CSI estimators in the simulation. Note that we compare networks under the same antenna diversity conditions, e.g. 2x2 ResNet with 2x2 SRGAN and never 2x1 ResNet with 2x2 SRGAN.

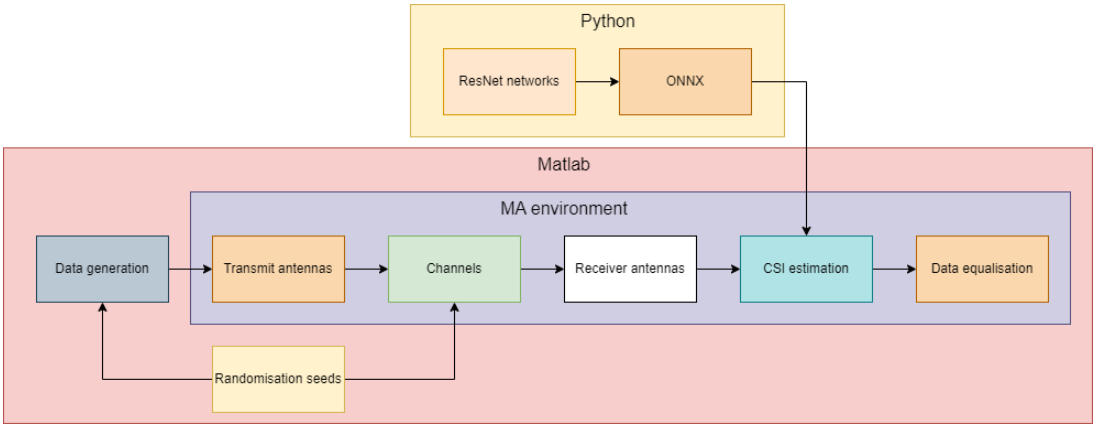


Figure 7.1: The interconnection of environments when importing trained models from Python to Matlab.

This simulation enables us to deploy trained networks in an MA environment in order to obtain the BER of symbols equalised using these CSI estimators.

7.3 Results

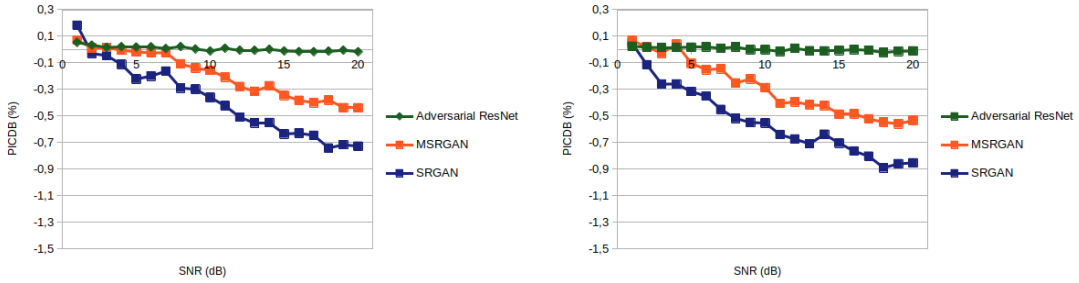
In this section, we compare the performance of networks that used different training methods in MA environments using different input sample sizes. Once again, we use the PICDB metric to compare our adversarially trained networks to our non-adversarially trained networks, as was done in Section 6.5. We first investigate receiver diversity, followed by transmitter diversity and then compare the results of our networks to LS and MMSE in these environments.

7.3.1 Receiver diversity results

We test the trained networks' CSI estimates in receiver diversity environments by implementing the EGC method with two and four receiving antennas.

Observing the performance of the 12x7 input sample size in Figure 7.2, we note that the adversarially trained networks do not perform similarly to what they have in the SISO

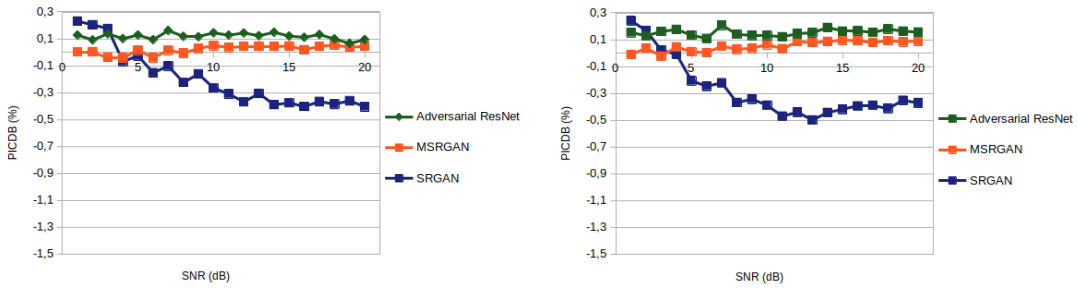
environment. Adversarial ResNet performs in a similar way to the non-adversarial ResNet. SRGAN and MSRGAN underperform non-adversarial ResNet, especially at higher SNRs, with MSRGAN outperforming SRGAN over much of the SNR range. We also note that with more receive antennas, the divide in relative performance between non-adversarial ResNet and the adversarial networks in the same diversity environment is wider than with fewer antennas.



(a) 1x2 (b) 1x4

Figure 7.2: PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 12x7 in a receiver diversity environment.

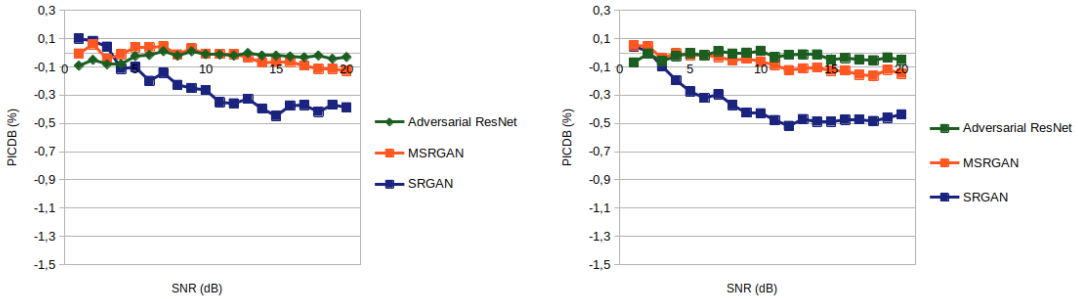
Figure 7.3 shows the performance of networks using 24x14 sample sizes. From the figure, we can see that adversarial ResNet outperforms non-adversarial ResNet in both antenna cases. Furthermore, MSRGAN also provides increased performance compared to non-adversarial ResNet at higher SNRs. Lastly, we see that SRGAN outperforms all networks at low SNRs, which can also be noted in Figure 7.2.



(a) 1x2 (b) 1x4

Figure 7.3: PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 24x14 in a receiver diversity environment.

Networks using 120x14 input sample sizes performance can be seen in Figure 7.4. This figure shows MSRGAN being competitive with non-adversarial ResNet at lower SNRs but dropping below non-adversarial ResNet at higher SNRs, but not as low as in Figure 7.3. Adversarial ResNet slightly underperforms non-adversarial ResNet over the entire SNR, along with SRGAN, which only outperforms all networks at low SNRs.



(a) 1x2 (b) 1x4

Figure 7.4: PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 120x14 in a receiver diversity environment.

We observe the average PICDB of these networks over the entire SNR range in a 1x2 receiver diversity simulation in Table 7.1. This table shows that non-adversarial ResNet

is the network of choice for most input sample sizes. The exception is Adversarial ResNet and MSRGAN using a 24x14 input sample size.

Table 7.1: Average PICDB over all SNRs of adversarially trained networks in a receiver diversity environment with two receiving antennas compared to a ResNet network.

Input sample size	Adversarial ResNet	SRGAN	MSRGAN
12x7	5.60e-3	-3.84e-1	-1.92e-1
24x14	**1.19e-1**	-2.04e-1	**1.98e-2**
120x14	-2.81e-2	-2.43e-1	-2.7e-2

Observing the performance of the same network in a 1x4 receiver diversity environment in Table 7.2 we see similar results to Table 7.1.

Table 7.2: Average PICDB over all SNRs of adversarially trained networks in a receiver diversity environment with four receiving antennas compared to a ResNet network.

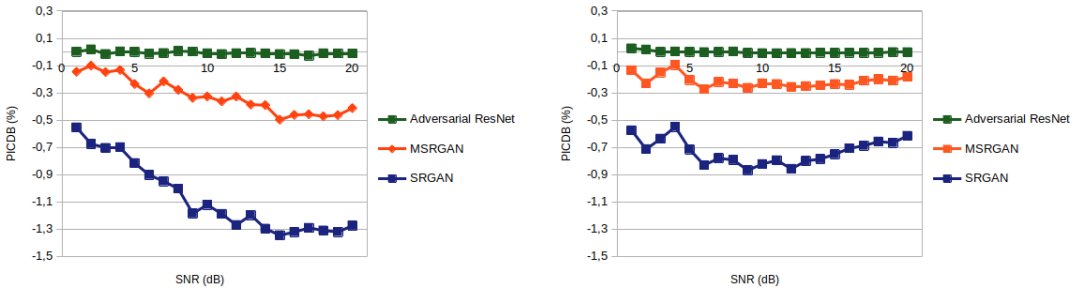
Input sample size	Adversarial ResNet	SRGAN	MSRGAN
12x7	2.58e-3	-5.44e-1	-2.92e-1
24x14	**1.53e-1**	-2.78e-1	**5.26e-2**
120x14	-2.41e-2	-3.55e-1	-7.03e-2

From Table 7.1 and Table 7.2, we observe the importance of input sample size, with some adversarial networks showing increased performance over the non-adversarial ResNet. We specifically focus on MSRGAN, which performs very poorly with small input sample sizes but shows increased performance with larger input sample sizes. Adversarial ResNet, on the other hand, shows increased performance over non-adversarial ResNet at an input sample size of 12x7 and 24x14 but a weaker performance at a 120x14 input sample size. Observing the networks over the entire SNR range we observe that adversarial networks tend to outperform non-adversarial ResNet at low SNR, especially as the input size increases.

7.3.2 Transmitter diversity results

We test the trained networks' CSI estimates in transmitter diversity environments by implementing the Alamouti method with one and two receiving antennas.

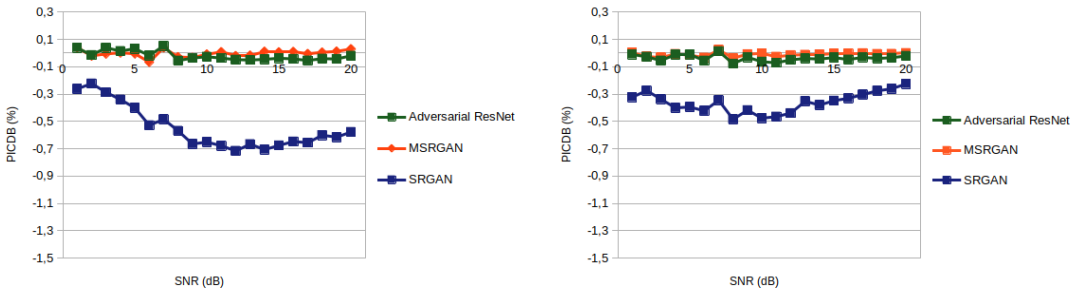
Observing Figure 7.5 in which an input sample size of 12x7 is used, we see that Adversarial ResNet is comparable with non-adversarial ResNet. MSRGAN and SRGAN underperform non-adversarial ResNet over the entire SNR range, with MSRGAN performing better than SRGAN. We also note that adding a receiving antenna improves the performance of the networks trained with feature extractors relative to non-adversarial ResNet.



(a) 2x1 (b) 2x2

Figure 7.5: PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 12x7 in a transmitter diversity environment.

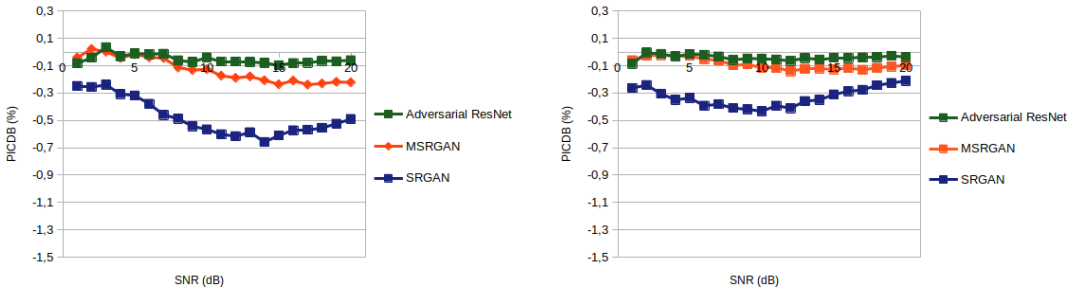
Figure 7.6 displays the PICDB of networks using an input sample size of 24x14. The figure shows that both Adversarial ResNet and MSRGAN are comparable to non-adversarial ResNet. SRGAN still underperforms the other networks, but again shows more comparable results when an additional receiving antenna is added. We also note that SRGAN increases relative performance after around 12 dB SNR.



(a) 2x1 (b) 2x2

Figure 7.6: PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 24x14 in a transmitter diversity environment.

The performance of networks using 120x14 input sample sizes can be seen in Figure 7.7. This figure shows that all adversarial networks underperform non-adversarial ResNet for most SNR ranges. We observe similar characteristics as in Figure 7.6 where SRGAN improves in performance and additional antennas improve relative performance between networks.



(a) 2x1 (b) 2x2

Figure 7.7: PICDB of the CSI generated by adversarially trained networks with respect to non-adversarially trained ResNet for an input sample size of 120x14 in a transmitter diversity environment.

Averaging the PICDB over the entire SNR range for the networks with a single receiving antenna in Table 7.3 shows that no network outperforms non-adversarial ResNet.

Table 7.3: Average PICDB over all SNRs of adversarially trained networks in a transmitter diversity environment with one receiving antenna compared to a ResNet network.

Input sample size	Adversarial ResNet	SRGAN	MSRGAN
12x7	-7.14e-3	-1.07	-3.22e-1
24x14	-1.93e-2	-5.48e-1	-1.85e-3
120x14	-5.32e-2	-4.79e-1	-1.31e-1

Similar results to Table 7.3 can be found in Table 7.4 where averaging the PICDB over the entire SNR range for the networks with two receiving antennas is shown.

Table 7.4: Average PICDB over all SNRs of adversarially trained networks in a transmitter diversity environment with two receiving antennas compared to a ResNet network.

Input sample size	Adversarial ResNet	SRGAN	MSRGAN
12x7	-1.26e-3	-7.30e-1	-2.15e-1
24x14	-3.65e-2	-3.63e-1	-1.05e-2
120x14	-3.95e-2	-3.30e-1	-8.83e-2

From Table 7.3 and Table 7.4, we note that networks respond differently to the increase in input sample sizes. For example, adversarial ResNet decreases relative performance as input sample size increases, while SRGAN shows an increase in relative performance under the same conditions. In addition, MSRGAN prefers an input sample size of 24x14 over the other input sample sizes. Observing the networks over the entire SNR range we see that relative performance between networks increases as the number of receiving antennas increases.

7.4 Discussion

This section discusses the results of Section 7.3 in which we applied adversarial and non-adversarial trained networks to SISO and MA environments. We first compare the relative results of our networks between SISO and MA environments, followed by a discussion of why adversarial training outperforms non-adversarial training in receiver diversity envi-

ronments.

7.4.1 SISO vs MA

Comparing the results of the adversarial and non-adversarial networks between SISO and MA environments, we observe inconsistent performance in terms of which network performs best of those trained. In Section 6.6 we observe that non-adversarially trained networks outperform all adversarial training methods in a SISO environment. Adversarial ResNet and MSRGAN are comparable when observing the entire SNR range, while SRGAN underperforms all training methods over the entire SNR range. This is, however, different in MA environments, with adversarial training methods outperforming non-adversarial training on some occasions, specifically in receiver diversity.

The alteration in which training method performs best as the deployment environment changes indicates that MA environments have different aspects of importance than SISO networks. This change in which training method performs best also indicates that adversarial training changes subtle features of the data in the estimated CSI compared to non-adversarial trained network estimations.

7.4.2 Receiver diversity discussion

We inspect the receiver diversity case as this is where adversarial methods outperform non-adversarial training, as seen in Figure 7.3. To find the difference in estimation between training methods, we observe cases in which an adversarially trained network correctly demodulated a symbol while non-adversarial ResNet did not. We find that these cases occur when one or more of the channels in the MA environment is experiencing deep attenuation.

It is found, using the evaluation dataset used in Chapter 6, that when one of the channels experiences deep attenuation, the final symbol of the ResNet estimated environments would be much larger than is expected of a symbol, while Adversarial ResNet and some-

times MSRGAN would be in the expected QI range. This occurrence of larger symbols was especially the case for the 24x14 input sample size. These "exploding symbols" are caused by the enhanced noise problem discussed in Section 3.7. For this reason, we investigate how the different training methods make estimations in deep attenuation situations.

In Figure 7.8 we display the number of CSI estimations made that are smaller than 5e-2 for both Q and I components using the SISO evaluation set over the entire SNR range. The figure shows that all networks predict more deep attenuation than is actually happening on the channel. Observing the number of estimations made by networks with 24x14 sample input sizes, we see that Adversarial ResNet and MSRGAN are lower than ResNet. We also observe that MSRGAN is the only training method that decreases the number of estimations as the input sample size increases.

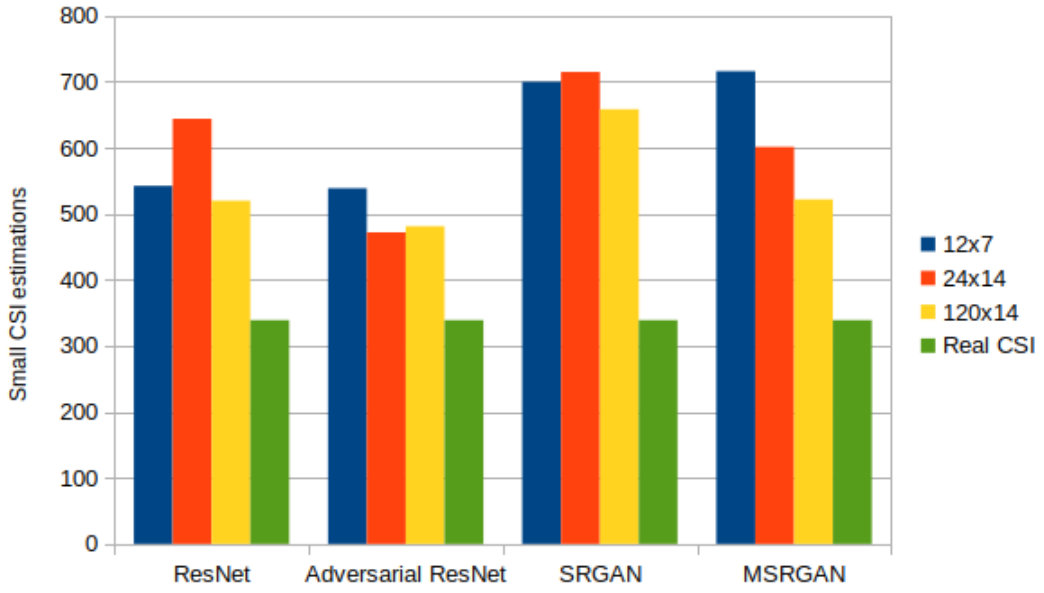


Figure 7.8: The number of CSI estimations made by networks that are smaller than 5e-2 for both Q and I components over the entire SNR range, using the test set.

Using the same dataset, we also investigate the number of equalised symbols with the absolute value of the largest Q and I of the expected symbols transmitted in a 16-QAM modulation scheme larger than 3, as this indicates our enhanced noise problem. We display the number of symbols in this range for each SNR in Figure 7.9 for the networks

with a 24x14 input sample size. The results in Figure 7.9 explain the behaviour of the networks in Figure 7.3 as the number of large equalised symbols correlates with network performance. We observe SRGAN having fewer large symbols at low SNRs than other networks before adversarial ResNet becomes the network with the least symbols over the rest of the SNR range. This coincides with SRGAN outperforming all networks at low SNRs in Figure 7.3.

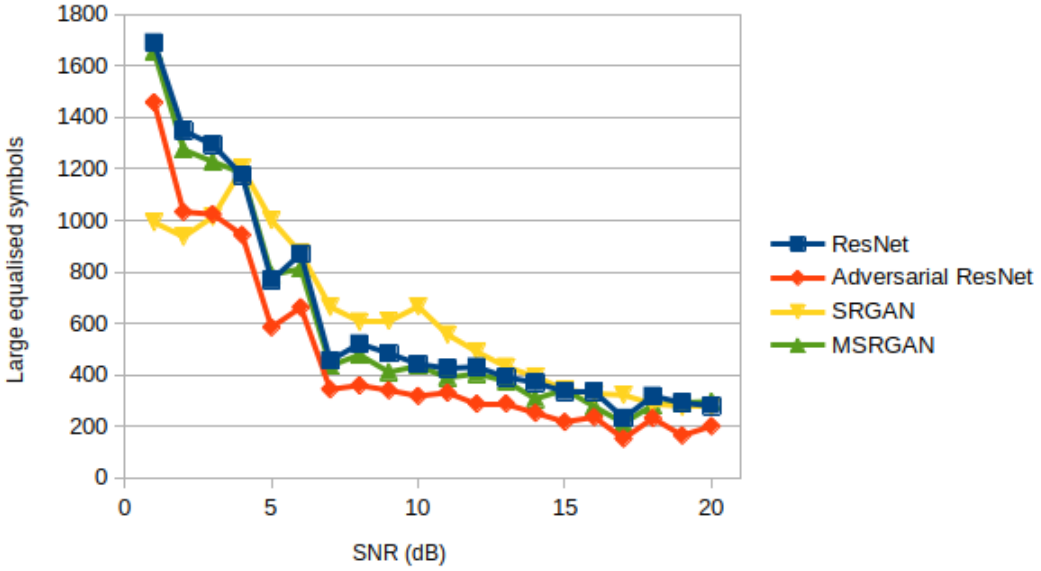


Figure 7.9: The number of symbols larger than 3 for both Q and I components equalised using different networks' CSI estimations over the SNR range, using the test set.

7.4.3 Enhanced noise problem in receiver diversity

The ability of networks to avoid the enhanced noise problem is an advantage in receiver diversity environments. By avoiding large equalised symbols, the final averaging of symbols becomes more accurate. For example, when observing Figure 7.10 we can see that when we have two channels, one of which has deep attenuation, we have one correctly demodulated symbol and one incorrectly demodulated symbol.

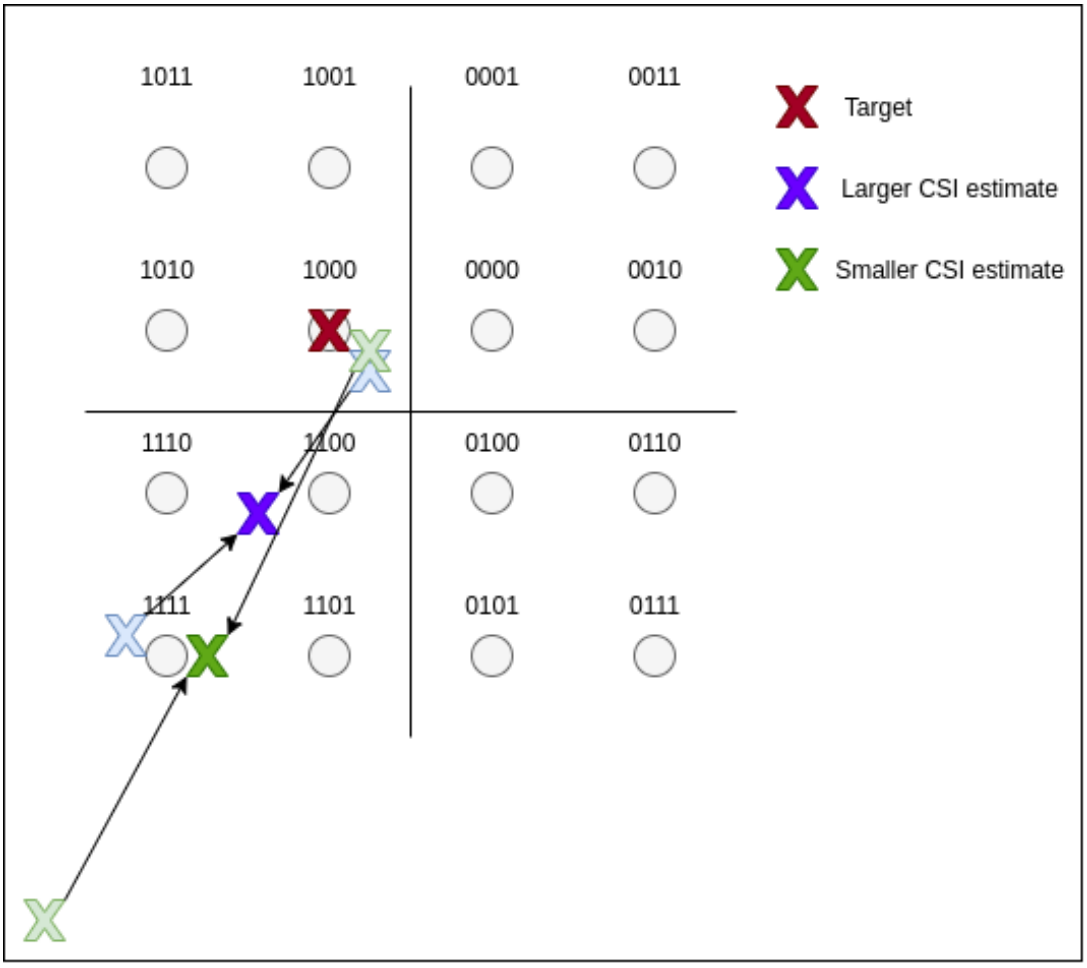


Figure 7.10: This figure indicates the effect small CSI estimations and enhanced noise can have in a receiver diversity system, by comparing symbols equalised using slightly larger and smaller CSI estimates.

The distance by which the incorrectly demodulated symbol is demodulated is, however, of importance and is dependent on the accuracy of the CSI estimation and the effect of noise enhancement. If the CSI estimation is small and noise enhancement takes place, the demodulated symbol can be placed much farther from the correct placement than a symbol with a larger CSI estimation with less enhanced noise effect. After averaging the two symbols, we can see that the equalised symbol with less noise enhancement produces a symbol with only one bit error compared to the alternative situation, which has multiple-bit errors.

Based on these reasons, we find that adversarially trained networks that avoid small esti-

mations perform better in receiver diversity environments. However, this estimation characteristic cannot be observed in SISO results because the placement of equalised symbols in deep attenuation situations does not consider the extent of the incorrect placement.

7.5 Benchmark comparison

From the results obtained in this chapter, non-adversarially trained ResNets with an input sample size of 120x14 perform best of all NNs. Using these networks, we compare the BER of equalised symbols using non-adversarial ResNet, LS and MMSE as CSI estimators.

In Figure 7.11, we see that ResNet slightly outperforms LS and MMSE over the entire SNR range in a SISO environment. Considering the results from Table 6.3, we also know that adversarial ResNet and MSRGAN would provide similar performances.

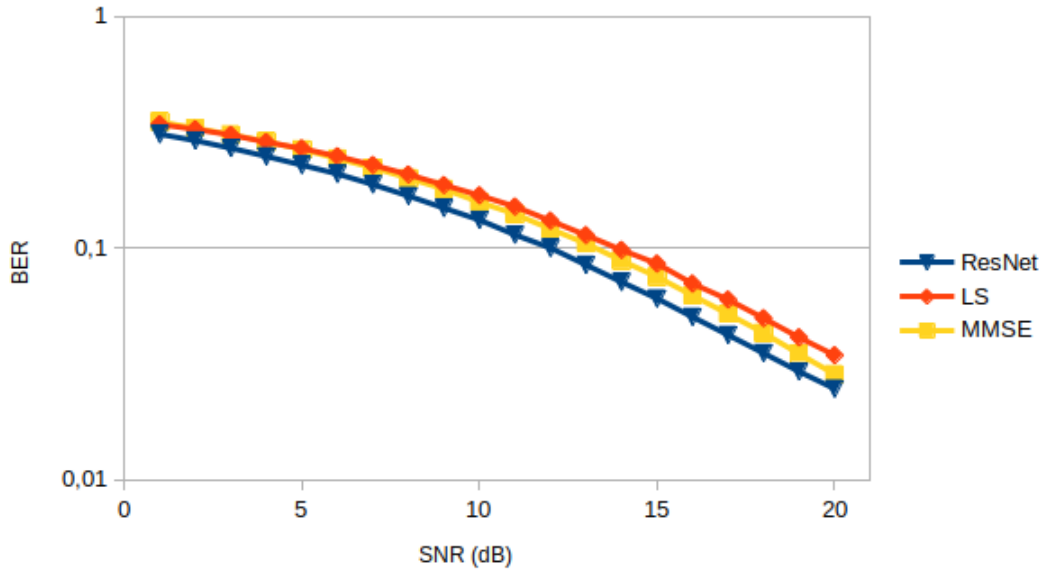


Figure 7.11: BER of 16-QAM equalised symbols in a SISO environment using ResNet, MMSE and LS CSI estimators over the SNR range, shown over the test set.

Figure 7.12 shows how ResNet outperforms the other estimation methods in a receiver diversity implementation with four receiving antennas. ResNet outperforms the other

methods by several dBs of SNR, over a range of SNR, before MMSE becomes competitive at 18 dB SNR. This figure displays the strengths of ResNet under high noise in MA environments.

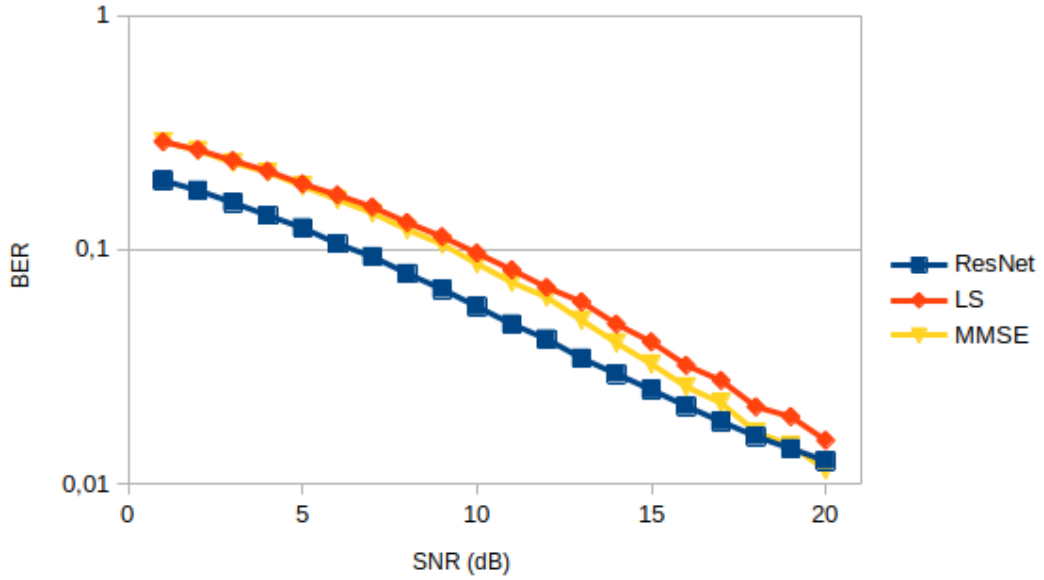


Figure 7.12: BER of 16-QAM equalised symbols in a receiver diversity environment with four receiver antennas using ResNet, MMSE and LS CSI estimators over the SNR range.

We see a continuation of these results in Figure 7.13 when the CSI estimators are applied to a transmitter diversity environment with two receiving antennas. In this figure, we see that ResNet outperforms the other methods by several SNRs over the entire range.

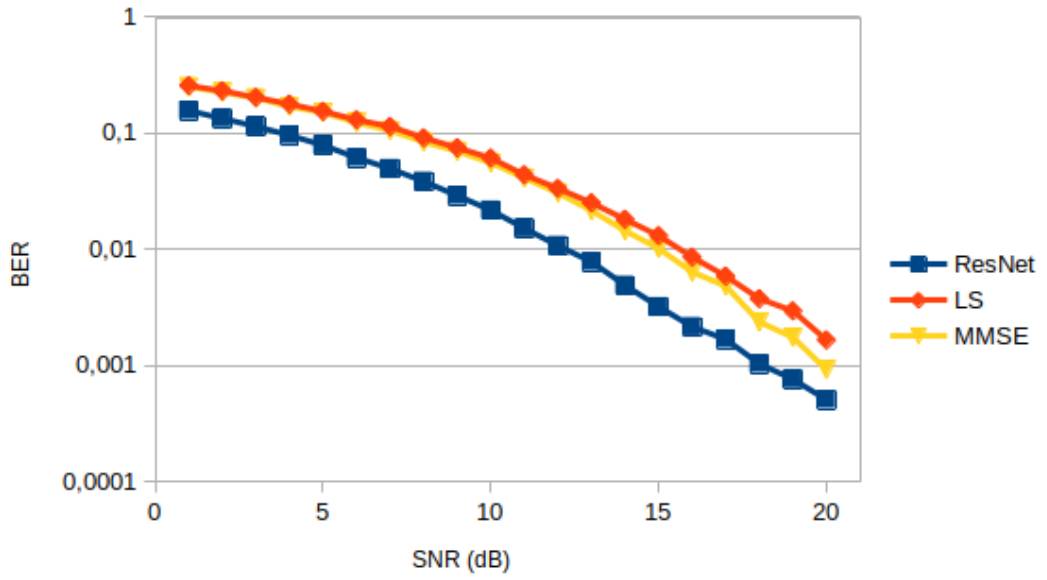


Figure 7.13: BER of 16-QAM equalised symbols in a transmitter diversity environment with two receiver antennas using ResNet, MMSE and LS CSI estimators over the SNR range.

All of these results indicate the ability of the ResNet architecture to outperform traditional LTE CSI estimation methods. The investigation of MA environments also shows that more relative performance is to be gained when antenna diversity is applied, even when using networks trained in SISO environments.

7.6 Conclusion

In this chapter, we investigate the ability of adversarial and non-adversarially trained networks in MA environments. We find that the performance of networks in a SISO environment is not indicative of performance in an MA environment, as adversarially trained networks outperformed non-adversarial trained networks in certain instances. Furthermore, based on the deployment environment and input sample size, the networks also behaved differently from the results obtained in the SISO environment in which they were trained.

Investigating the cases in which adversarial training outperformed non-adversarial training, we find that receiver diversity implementations are more sensitive to the position of equalised symbols in the QI plane. Adversarial ResNet and MSRGAN using input sample sizes of 24x14 could take advantage of this characteristic to outperform ResNet by reducing the number of small CSI estimations. This characteristic was brought to light by adversarial training and was not detectable in the SISO evaluation results.

Finally, we evaluate the best performing networks to LTE CSI estimators over SISO and MA environments and find that ResNet outperforms all baselines discussed in Section 3.7. ResNet outperforms baselines by several dBs of SNR, in some cases up to 4 dB, which equals doubling the transmitting signal strength. In addition, it is noted that non-adversarial ResNet using a 120x14 input sample size was the best performing network in all environments. We expect larger sample sizes to keep increasing performance, but due to the computational expense of networks with large input sizes we do not investigate this further.

Some of these results confirm the claims made by Zhoa *et al.* [16], indicating that adversarial training can increase the BER of CSI estimators in certain conditions. We do, however, indicate that adversarial training does not always increase performance and, in certain conditions, can significantly decrease the BER obtained by CSI estimators.

Chapter 8

Conclusion

In this chapter we discuss key findings and the implications of these findings. We also identify research questions for future work.

8.1 Introduction

The main goal of this study is, as discussed in Chapter 1, to investigate adversarially trained networks in MA environments. In this chapter, we conclude this study by comparing the work done to the objective set in Section 1.3. These objectives are:

- Investigate channel impairments found in wireless communication environments by simulating channels with a variety of impairments to generate CSI datasets.
- Make use of different data features and dataset compositions to investigate how deep learning networks trained on different CSI datasets compare with each other.
- Apply adversarially and non-adversarially trained networks to the CSI estimation task.

- Investigate adversarial network implementations in MA environments for the CSI estimation task.

We do this by reviewing key findings, observations and summarising the contributions made by this study. Finally, we propose research questions for future work.

8.2 Key findings

In this section, we discuss the key findings of this study. All of these findings are made using channels with an EVA delay profile and 50 Hz of Doppler spread using ResNet as a base architecture that is trained using different adversarial training methods to estimate CSI over a range of SNRs.

- We found that increasing the range of SNRs represented by samples in the training dataset increased the ResNets' generalisation ability in a SISO environment. Using a single network trained on a distributed dataset of three separate SNRs we obtain performance comparable with an implementation that uses three separate networks each trained on a dataset evaluating a single SNR value. We thus obtain comparable CSI estimation performance over a range of SNRs at a third of the computational training and deployment cost by adapting the training dataset.
- We found that adversarial training does not provide any consistent advantage over the traditional training method in a SISO environments. Adversarial ResNet and MSRGAN, however, provide similar BER than non-adversarially trained ResNet at certain SNRs, thus showing that they can be capable CSI estimators in SISO environments. When considering the entire SNR range we observe that non-adversarially trained networks provide on average, over the SNR range, the best performance of all tested networks.
- In investigating MA environments we found that adversarial ResNet and MSRGAN with 24x14 sample input sizes refrain from estimating small CSI values compared

to ResNet and SRGAN. CSI estimations that have fewer small-valued estimations provided better results in receiver diversity environments due to avoiding the noise enhancement problem which creates large QI points when equalised. This difference in CSI estimations could not be detected in the SISO environment using standard evaluation metrics.

8.3 Observations

In this section, we discuss the observations made in this study that are not critical to the problem statement but nevertheless played an important role in the study. We start with the observations made when investigating our dataset using MLPs and CNNs:

- We find that structural hyperparameters of an MLP architecture can play a role in the network's ability to estimate CSI. Structural hyperparameters are adjusted and are found to potentially increase network performance as the network's size is increased.
- From the investigation into feature representation of data samples we find that MLPs have difficulty interpreting CSI data that is non-Euclidean, especially if noise is introduced to the input samples.
- Batch normalisation was found to produce smooth hyperparameter search grid results when training MLPs on noisy data. It was, however, found that MLPs could interpret simple non-Euclidean data better if batch normalisation was removed from MLPs.
- When implementing the ResNet architecture using the developed dataset it was found that the spatial-temporal abilities of CNNs outperformed MLPs, which in turn was comparable with the LTE baselines.

Using the ResNet architecture proposed by Zhao *et al.* [16] and the EENDD developed

in this study we observe the following when investigating several adversarial training methods and MA environments:

- Increasing the input sample size of the ResNet architecture results in an increase in performance. This is due to the increased number of subcarriers available to the ResNet, allowing for a more accurate transition between slots.
- Deploying networks trained in a SISO environment in an MA environment has varying results that are not consistent with SISO performance, such as:
  - Adversarial networks that use feature extractors in training perform considerably worse at small input sample sizes, especially when compared to SISO environment results.
  - Networks with larger input sample sizes that are adversarially trained can outperform non-adversarially trained networks in receiver diversity environments.
  - In some instances SRGAN networks can outperform other networks at low SNRs.
- We find that using increased input sample sizes increases the performance of all networks.
- The non-adversarially trained ResNet with an input sample size of 120x14, outperforms all other networks and benchmarks in this study in all antenna environments.

8.4 Contributions

In this study, we evaluate the performance of a set of NNs, trained in a SISO environment, in MA environments and make findings that contribute to the knowledge of CSI estimation problem using deep learning. These contributions are:

- A study into the composition of CSI generated datasets for deep learning that include aspects such as:

  - Channel impairments on final equalised data
  - Feature representation of data samples
  - Dataset size
  - SNR distribution of sample amounts in the dataset
  - Input sample sizes

- We created an environment for LTE data generation and MA environment model analysis in Matlab® that allows for training and exporting deep learning models from Python to Matlab®.
- The introduction of an adversarial training architecture that makes use of a feature extractor without impacting network performance.
- Indications that in certain MA environments adversarial training can increase network performance even if this is not evident in the training environment.

8.5 Future work

Based on the findings of the study, we identify additional research questions:

- To what extent can input sample sizes be increased before BER stops increasing?
- Can the performance of deep learning networks for the CSI estimation task be increased by training networks in MA environments instead of the SISO environment?
- Are there alternative deep learning architectures that are more responsive to adversarial training than ResNet for CSI estimation?
- Are there further MA methods that show increased performance when using CSI estimators that have been trained adversarially and have estimation features that are unseen in the training environment?

8.6 Conclusion

CSI estimation plays an important role in the wireless communication systems we use daily, especially with new standards that transmit over increasingly difficult-to-estimate channels. With adversarial training methods obtaining state-of-the-art results in computer vision tasks we investigate the use of this training method on CSI estimation that has previously found success from computer vision methods. This study presents the use of adversarial training to increase the performance of deep learning-based CSI estimators without increasing the deployment cost of networks in certain antenna diversity environments. In addition, the study shows a clear mismatch between the environments CSI estimators are trained and tested in. We hope that the work presented in this study leads to future work in deep learning-based CSI estimators using adversarial training, specifically in MA environments.

Bibliography

[1] Y. Liu, Z. Tan, H. Hu, L. J. Cimini, and G. Y. Li, "Channel estimation for OFDM," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1891–1908, 2014.

[2] K. Hassan, M. Masarra, M. Zwingelstein, and I. Dayoub, "Channel estimation techniques for millimeter-wave communication systems: Achievements and challenges," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 1336–1363, 2020.

[3] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[5] C. Ledig, L. Theis, F. Huszár, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[6] B. Ghosh, I. K. Dutta, M. Totaro, and M. Bayoumi, "A survey on the progression and performance of generative adversarial networks," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1–8. DOI: 10.1109/ICCCNT49239.2020.9225510.

[7] Y. Yang, Y. Li, W. Zhang, F. Qin, P. Zhu, and C.-X. Wang, "Generative-adversarial-network-based wireless channel modeling: Challenges and opportunities," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 22–27, 2019.

[8] H. Ye, G. Y. Li, B.-H. F. Juang, and K. Sivanesan, "Channel agnostic end-to-end learning based communication systems with conditional GAN," in *2018 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2018, pp. 1–5.

[9] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," *IEEE Communications Letters*, vol. 23, no. 4, pp. 652–655, 2019. DOI: 10.1109/LCOMM.2019.2898944.

[10] M. v. Lier, A. Balatsoukas-Stimming, H. Corporaal, and Z. Zivkovic, "OPTCOM-NET: Optimized neural networks for low-complexity channel estimation," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9149049.

[11] Y. Yang, F. Gao, X. Ma, and S. Zhang, "Deep learning-based channel estimation for doubly selective fading channels," *IEEE Access*, vol. 7, pp. 36 579–36 589, 2019. DOI: 10.1109/ACCESS.2019.2901066.

[12] Y. Liao, Y. Hua, X. Dai, H. Yao, and X. Yang, "ChanEstNet: A deep learning based channel estimation for high-speed scenarios," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8761312.

[13] T. Wang, C.-K. Wen, S. Jin, and G. Y. Li, "Deep learning-based CSI feedback approach for time-varying massive MIMO channels," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 416–419, 2019. DOI: 10.1109/LWC.2018.2874264.

[14] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," 2019. arXiv: 1812.04948.

[15] T. Hu, Y. Huang, Q. Zhu, and Q. Wu, "Channel estimation enhancement with generative adversarial networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 145–156, 2021. DOI: 10.1109/TCCN.2020.3013257.

[16] S. Zhao, Y. Fang, and L. Qiu, "Deep learning-based channel estimation with SR-GAN in OFDM systems," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1–6. DOI: 10.1109/WCNC49053.2021.9417242.

[17] J. Zyren and W. McCoy, "Overview of the 3GPP long term evolution physical layer," *Freescale Semiconductor, Inc., white paper*, vol. 7, pp. 2–22, 2007.

[18] *Evolved universal terrestrial radio access E-UTRA; base station BS conformance testing*, 3GPP TS 36.141, Jul. 2018.

[19] A. Oosthuizen, M. H. Davel, and A. Helberg, "Exploring CNN-based automatic modulation classification using small modulation sets," Southern Africa Telecommunication Networks and Applications Conference, 2021, pp. 20–24.

[20] A. Oosthuizen, M. H. Davel, and A. Helberg, "Multi-layer perceptron for channel state information estimation: Design considerations," Southern Africa Telecommunication Networks and Applications Conference, 2022, pp. 94–99.

[21] A. Oosthuizen, M. H. Davel, and A. Helberg, "Adversarial training for channel state information estimation in lte multiantenna systems," *Communications in Computer and Information Science*, vol. 1734, Accepted for publication.

[22] Ericsson, Ericsson mobility report, Available: https://www.ericsson.com/assets/local/mobility-report/ documents/2017/ericsson-mobility-report-june-2017.pdf, 2017.

[23] D. J. Griffiths, *Introduction to electrodynamics*. Pearson, 2013, ISBN: 9781108420419.

[24] B. Forouzan, *Data Communications and networking*. McGraw-Hill, 2013, ISBN: 9789814577519.

[25] G. Stuber, J. Barry, S. McLaughlin, Y. Li, M. Ingram, and T. Pratt, "Broadband MIMO-OFDM wireless communications," *Proceedings of the IEEE*, vol. 92, no. 2, pp. 271–294, 2004. DOI: 10.1109/JPROC.2003.821912.

[26] Y. Wu and W. Y. Zou, "Orthogonal frequency division multiplexing: A multi-carrier modulation scheme," *IEEE Transactions on Consumer Electronics*, vol. 41, no. 3, pp. 392–399, 1995.

[27] Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *MIMO-OFDM wireless communications with MATLAB*. IEEE Press, 2011, ISBN: 9780470825617.

[28] A. Grami, "Chapter 12 - Wireless Communications," in *Introduction to Digital Communications*, A. Grami, Ed., Boston: Academic Press, 2016, pp. 493–527, ISBN: 978-0-12-407682-2. DOI: https://doi.org/10.1016/B978-0-12-407682-2.00012-0.

[29] I. Glover and R. Atkinson, "1 - Overview of wireless techniques," in *Wireless MEMS Networks and Applications*, ser. Woodhead Publishing Series in Electronic and Optical Materials, D. Uttamchandani, Ed., Woodhead Publishing, 2017, pp. 1–33, ISBN: 978-0-08-100449-4. DOI: https://doi.org/10.1016/B978-0-08-100449-4.00001-4.

[30] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke, "The IEEE 802.11 universe," *IEEE Communications Magazine*, vol. 48, no. 1, pp. 62–70, 2010.

[31] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of LoRaWAN for IoT: From technology to application," *Sensors*, vol. 18, no. 11, p. 3995, 2018.

[32] *E-UTRA, base station (BS) radio transmission and reception*, 3GPP, May 2008.

[33] M. MOQBEL, W. Wangdong, and Z. al-marhabi, "MIMO channel estimation using the LS and MMSE algorithm," *IOSR Journal of Electronics and Communication Engineering*, vol. 12, pp. 13–22, Jan. 2017. DOI: 10.9790/2834-1201021322.

[34] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on selected areas in communications*, vol. 16, no. 8, pp. 1451–1458, 1998.

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, pp. 139, 145, 151–152, 167–170, 192–194, 224–226, 251, 254, http://www.deeplearningbook.org.

[36] S. Pouyanfar, S. Sadiq, Y. Yan, *et al.*, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[37] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and electronics in agriculture*, vol. 147, pp. 70–90, 2018.

[38] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100 379, 2021.

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, pp. 139, 145, 167–170, 192–194, http://www.deeplearningbook.org.

[40] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[41] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, IEEE, 2017, pp. 1–6.

[42] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[43] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*. 2020, `https://d2l.ai`.

[44] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2019, preprint on webpage at `https://arxiv.org/abs/1803.08375`.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the ICVPR*, 2015, pp. 1026–1034.

[46] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[47] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[48] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[49] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, pp. 151–152, `http://www.deeplearningbook.org`.

[50] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Springer, 2010, pp. 177–186.

[51] T. Tieleman, G. Hinton, *et al.*, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[52] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, Dec. 2014, arXiv preprint arXiv:1412.6980.

[53] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, pp. 228–280, `http://www.deeplearningbook.org`.

[54] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[55] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[56] M. Mirza and S. Osindero, *Conditional generative adversarial nets*, 2014. arXiv: `1411.1784`.

[57] Z. Jiang, S. Chen, A. F. Molisch, R. Vannithamby, S. Zhou, and Z. Niu, "Exploiting wireless channel state information structures beyond linear correlations: A deep learning approach," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 28–34, 2019.

[58] N. Soltani, H. Cheng, M. Belgiovine, *et al.*, "Neural network-based OFDM receiver for resource constrained IoT devices," *arXiv preprint arXiv:2205.06159*, 2022.

[59] C. Luo, J. Ji, Q. Wang, X. Chen, and P. Li, "Channel state information prediction for 5G wireless communications: A deep learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 227–236, 2018.

[60] W. Freeman, T. Jones, and E. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002. DOI: `10.1109/38.988747`.

[61] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016. DOI: `10.1109/TPAMI.2015.2439281`.

[62] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017. DOI: 10.1109/TIP.2017.2662206.

[63] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[64] H. Zimmermann, "OSI reference model-the ISO model of architecture for open systems interconnection," *IEEE Transactions on communications*, vol. 28, no. 4, pp. 425–432, 1980.

[65] T. Wang, C.-K. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, "Deep learning for wireless physical layer: Opportunities and challenges," *China Communications*, vol. 14, no. 11, pp. 92–111, 2017.

[66] H.-C. Tsai, C.-J. Chiu, P.-H. Tseng, and K.-T. Feng, "Refined autoencoder-based CSI hidden feature extraction for indoor spot localization," in *2018 IEEE 88th vehicular technology conference (VTC-Fall)*, IEEE, 2018, pp. 1–5.

[67] D. Erdogmus, D. Rende, J. C. Principe, and T. F. Wong, "Nonlinear channel equalization using multilayer perceptrons with information-theoretic criterion," in *Neural Networks for Signal Processing XI: Proceedings of the 2001 IEEE Signal Processing Society Workshop (IEEE Cat. No. 01TH8584)*, IEEE, 2001, pp. 443–451.

[68] E. Ahmed, A. Saint, A. E. R. Shabayek, *et al.*, "A survey on deep learning advances on different 3D data representations," *arXiv preprint arXiv:1808.01462*, 2018.

[69] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" *Advances in neural information processing systems*, vol. 31, 2018.

[70] A. Mehmood and W.-A. Cheema, "Channel estimation for LTE downlink," Ph.D. dissertation, Blekinge Institute of Technology, 2009.

Appendix A

Supplementary content

A.1 Appendix: Chapter 3

In this section, we validate our simulation using published results that use similar methods and systems. By making this comparison we ensure that our simulation code base can be used as a creditable data source and test bed. This comparison also confirms the validity of all physical layer blocks and their implementation as presented in Figure 3.1. In addition, we will be validating the following implementations that were not directly imported from an accredited Matlab® package:

- Resource block population
- Pilot symbol insertion
- LS CSI estimator
- Symbol equalisation using CSI

We compare our simulated results with those of Mehmood et al. [70] as their simulation setup is similar to that constructed in this study. We simulate a physical layer LTE transmission in a SISO environment using the conditions described by Mehmood et al. [70]:

- 4-QAM modulation
- EVA delay profile
- 120 Hz Doppler spread
- Resource block size of 900x14
- LS CSI estimator

By adapting the simulation created in Chapter 3 to these conditions, we compare the BER over a range of 0 to 20 dB SNR to the results of Mehmood et al. [70]. To ensure consistency, each SNR is simulated until at least 200 bit errors are found. In Figure A.1 we see that the simulation developed in this study produces results comparable to Mehmood et al. [70]. We include error bars in this figure representing a 95% confidence interval.

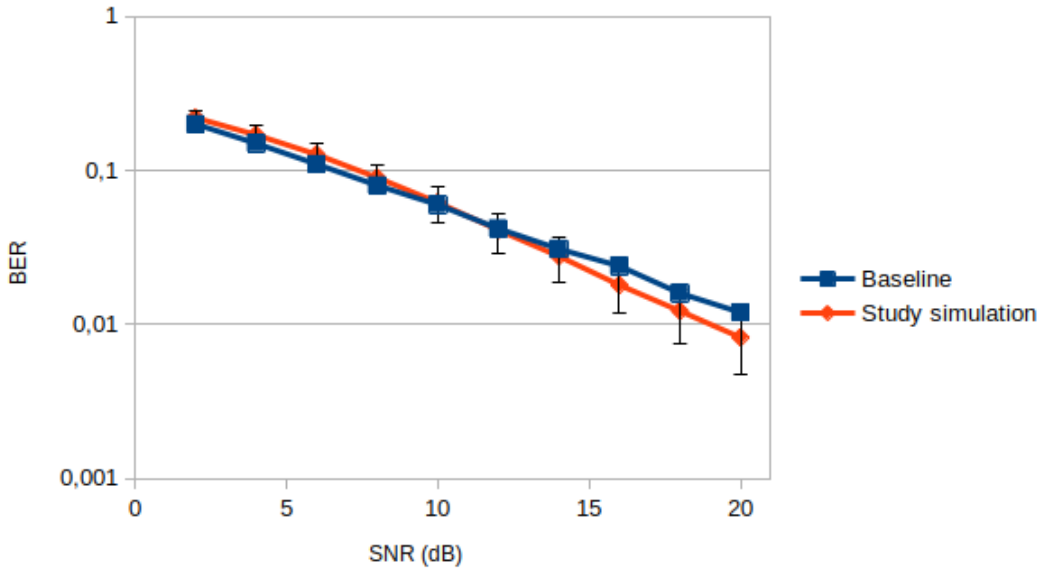


Figure A.1: A BER comparison between the results of Mehmood et al. [70] for a SISO channel and the simulation created for this study. The error bars represent a 95% confidence interval.

This comparison thus validates all implementations which trustworthy sources have not provided.

A.2 Appendix: Chapter 5

In order to ensure that appropriate hyperparameters are used in the training of our ResNets, we start with an initial hyperparameter sweep. This sweep is extended if the best performing hyperparameters were found at the edge of the hyperparameter range.

If hyperparameter search grids were extended for a network, the search grid is also extended for all other networks using the same input sample size. We do this to ensure that the trained networks are comparable to other networks using the same input sample size. Tables A.1, A.2 and A.3 each refer to the final hyperparameter range used for training different input sample sizes.

Table A.1: Hyperparameter ranges used for the grid search for an input sample size of 12x7.

Hyperparameter	Value
Generator Learning Rate (LR)	1e-3, 1e-4, 1e-5
Discriminator LR	1e-3, 1e-4, 1e-5
Discriminator ratio	0, 1e-2, 1e-3, 1e-4

Table A.2: Hyperparameter ranges used for the grid search for an input sample size of 24x14.

Hyperparameter	Value
Generator LR	1e-3, 1e-4, 1e-5
Discriminator LR	1e-2, 1e-3, 1e-4, 1e-5, 1e-6
Discriminator ratio	0, 1e-2, 1e-3, 1e-4

Table A.3: Hyperparameter ranges used for the grid search for an input sample size of 120x14.

Hyperparameter	Value
Generator LR	1e-3, 1e-4, 1e-5, 1e-6
Discriminator LR	1e-3, 1e-4, 1e-5, 1e-6
Discriminator ratio	0, 1e-2, 1e-3, 1e-4

A.3 Confidence

A.3.1 BER

In order to ensure that measured BER approaches actual BER, the number of bits tested needs to approach infinity. We can, however, make use of confidence intervals to ensure that the measured BER is within a certain confidence level of the actual BER. In order to do this, we need to test a set number of bits for a given number of errors, or BER. Using Equation A.1[1] we can calculate the confidence level of our BER measurement based on the number of errors and tested bits.

$$CL = 1 - e^{-\text{Total bits}*\text{BER}} \tag{A.1}$$

We use Equation A.1 to test the confidence level of the lowest measured BER in our results. The lowest BER can be found in Figure 7.13 and is 7e-4 so we round down to a BER of 1e-5. All test sets and simulations used in Chapters 6 and 7 transmit 672 000 bits that are used to calculate the BER. Using these metrics in Equation A.2 we obtain a confidence level of 99.9%.

$$CL = 1 - e^{-672000*1e-5} CL = 9.99e - 1 \tag{A.2}$$

This means that measured BER statistics obtained are accurate to actual BER, and enough bits are tested for BER statistics given the BER levels we are working with.

A.3.2 Models

Knowing that the BER statistics obtained by our models are accurate we move on to model confidence. We train three models, each using different initialisation seeds for each training architecture in Chapters 6 and 7. Results are reported for a specific training architecture on the average BER obtained between these three networks.

[1] `https://edadocs.software.keysight.com/kkbopen/how-do-i-measure-the-bit-error-rate.-ber-to-a-given-confidence-level-on-the-j-bert-m8020a-and-the-m8040a-high-performance.-bert-588276182.html`

To ensure that the reported results are accurate, we test the most noteworthy result of this study which is the receiver diversity PICDB results of networks with an input sample size of 24x14 shown in Figure 7.3. We test these results by adding error bars showing the standard deviation between the obtained BER of the three networks. Figure A.2 indicates these statistics for a receiver diversity environment with 2 receiving antennas, while Figure A.3 shows 4 receiving antennas.

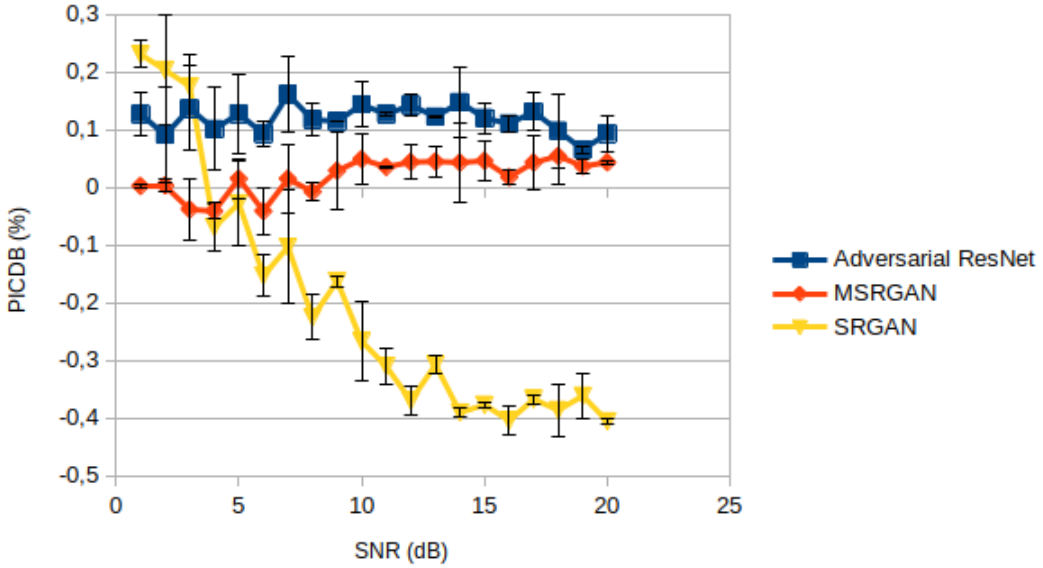


Figure A.2: PICDB, with standard deviations, of the CSI generated by adversarially trained networks compared to non-adversarially trained ResNet for an input sample size of 24x14 in a receiver diversity environment with 2 receiving antennas.

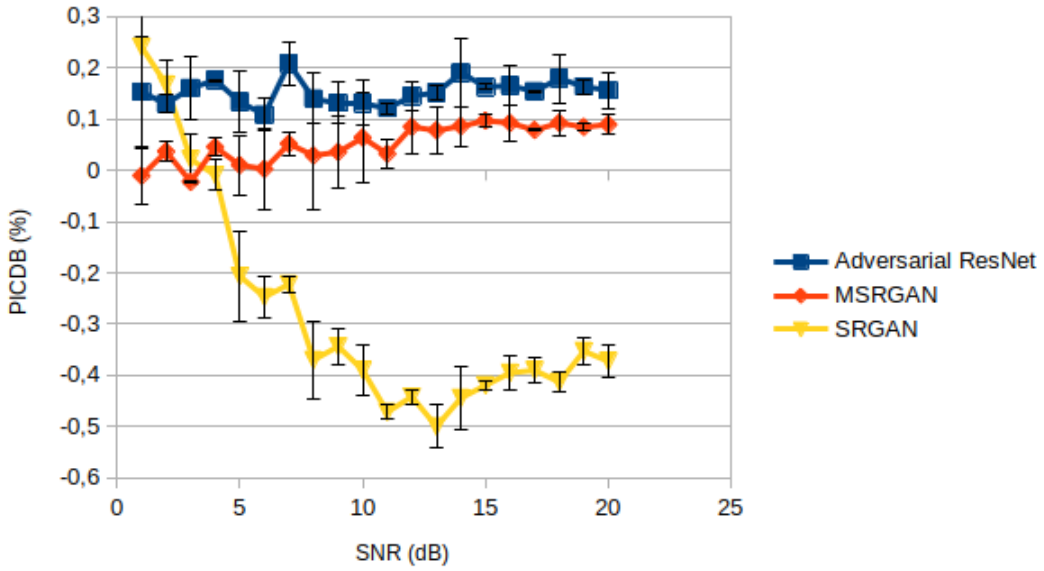


Figure A.3: PICDB, with standard deviations, of the CSI generated by adversarially trained networks compared to non-adversarially trained ResNet for an input sample size of 24x14 in a receiver diversity environment with 4 receiving antennas.

From the figures we see that at higher SNRs there is rarely an overlap between the adversarial ResNet and MSRGAN with the 0% line. These results indicate that our previous observations made in Section 7.4.2 that adversarially trained networks can outperform traditionally trained networks in certain conditions still hold.

A.4 Published work

A.4.1 Exploring CNN-Based Automatic Modulation Classification Using Small Modulation Set

Exploring CNN-Based Automatic Modulation Classification Using Small Modulation Sets

Andrew Oosthuizen*†, Marelie H. Davel*†, Albert Helberg*
*School of Electrical, Electronic and Computer Engineering, North-West University
†Centre for AI Research (CAIR), South Africa
aj.oosthuizen.ao@gmail.com
marelie.davel@nwu.ac.za
albert.helberg@nwu.ac.za

***Abstract*—We investigate the effect of a reduced modulation scheme pool on a CNN-based automatic modulation classifier. Similar classifiers in literature are typically used to classify sets of five or more different modulation types [1] [2], whereas our analysis is of a CNN classifier that classifies between two modulation types, 16-QAM and 8-PSK, only. While implementing the network, we observe that the network's classification accuracy improves for lower SNR instead of reducing as expected. This analysis exposes characteristics of such classifiers that can be used to improve CNN classifiers on larger sets of modulation types. We show that presenting the SNR data as an extra data point to the network can significantly increase classification accuracy.**

***Index Terms*—Automatic Modulation Classification, In-phase and Quadrature-phase (I/Q) symbols, Deep learning**

I. Introduction

In this paper we investigate a deep learning based approach to automatic modulation classification (AMC). AMC is used in the telecommunications field to identify transmission modulation schemes without this information being explicitly communicated between transmitters and receivers. AMC reduces overhead in communication and allows for effective switching between modulation schemes in cognitive radio applications. In the past, AMC has been implemented with statistical [3] and machine learning methods, such as clustering [4] and support vector machines [5]. In recent years deep learning architectures such as multilayer perceptrons (MLPs) and convolutional neural networks (CNNs) have been applied to the problem and have shown better performance over the more traditional approaches with regard to both accuracy and speed [6].

This paper investigates classification behaviour of deep neural networks on modulation types under additive white Gaussian noise (AWGN), by classifying between two modulation schemes, both varying in type and order. By using a reduced modulation pool for classification, we are able to better understand how a CNN interacts with an AMC task.

II. Related work

There exist several methods to approach AMC using deep learning models [7]. Most approaches supply some constellation data obtained from raw signal data to a neural network. How the constellation data is presented to the neural network does, however, vary depending on the method. Popular methods include presenting the quadrature and in-phase data points of constellation diagrams in a $2 \times N$ array, where N is the number of data points [2], or presenting the constellation plots as images [8]. The last method often contains several stages of feature extraction and pre-processing before the data is presented to the network.

CNNs are typically used for AMC problems [7] and function by presenting the input data to convolutional layers. The convolutional layers extract features from the data by making use of filters, also known as kernels. After feature extraction, the convolutional layers are flattened and passed to dense layers that make use of the previously extracted features to perform classification [9], [10].

For our study we use a CNN structure, similar to the network used by Yongshi et al. [2], that receives constellation data points rather than images as inputs. The reason for this is to simplify the investigation of the neural network, since preprocessing and presentation of image data to CNNs add extra levels of analysis to the process. We make use of 16th order quadrature amplitude modulation (16-QAM) and 8th order phase-shift keying (8-PSK) modulation schemes as input, as both the order and method of modulation differ between the two.

III. Experimental setup

A. Data

The data is presented as complex values of the signal constellation in the I/Q plane generated using Matlab version 2020b. A random bit stream source is modulated in baseband using one of two modulation types (8-PSK or 16-QAM). The modulated data is sent over an additive white Gaussian noise (AWGN) channel with varying normalised signal-to-noise (SNR) ratios (E_b/N_o) with an average signal power of 1W over 1Ω. The complex valued channel symbols are then grouped into samples containing 1024 constellation points of each symbol's in-phase and quadrature component. Thus, each sample consists of 1024 32-bit real and 1024 32-bit imaginary data points to create a 2×1024 sized data set.

The SNR, E_b/N_o, is discretely stepped over the range of -15 dB to 5 dB in 1 dB increments to create training, validation

and evaluation sets respectively. The number of generated data samples per E_b/N_o is listed in Table I.

TABLE I
NUMBER OF DATA SAMPLES GENERATED PER E_B/N_O OF A MODULATION SCHEME, AS WELL AS THE TOTAL SET SIZE OVER 21 SNR RANGE.

Modulation	Train	Validation	Test
8-PSK	1 000	500	1 000
16-QAM	1 000	500	1 000
Total set size	42 000	21 000	42 000

The training and validation sets are used in the training process as described below, while the evaluation sets are kept separate to evaluate the performance per E_b/N_o level.

B. Baseline architecture

The classifier architecture is based on that of Yongshi et al. [2], but with fewer nodes in the hidden dense layer. The hidden dense layer is reduced to 100 nodes, as the number of modulation types to classify has been reduced. This change is made to improve the training time and throughput of the network. The network consists of two convolutional layers that are ReLu-activated [11], makes use of batch normalisation, has no padding and a stride of 1. A max pooling layer, with a stride of 2, is placed between the convolutional layers to reduce the complexity of the network. After the convolutional layers a linear layer with 100 nodes is placed, followed by the classification layer (also a linear layer) with 2 outputs [2].

C. Training protocol

The same training protocol is followed for all networks. Networks are trained with the Adam [12] optimiser using a cross-entropy loss function. Adam is selected for its ability to adapt the learning rate of different parameters, and cross-entropy loss is used for its good performance in classification problems [9], [10]. Since ReLU activation functions are used, the weights of the network are initialised using a uniform Kaiming initialisation [13].

The following hyperparameters are optimised: learning rate, batch size, and weight decay (L2 penalty). We selected these hyperparameters, as preliminary tests showed that they have noticeable effects on the model's performance. Hyperparameter tuning is performed using grid searches on the predefined CNN architecture, by comparing the networks' results on the validation data set. The grid search is performed over different learning rates $\{0.01, 0.001, 0.0001\}$, batch sizes $\{32, 512\}$, weight decay values $\{0, 0.001, 0.01\}$ and 3 random initialisation seeds. The initialisation seeds are used to ensure a particularly strong or weak network initialisation does not affect the results. We also make use of a grid search over the architecture by varying the convolution kernel width $\{4, 16, 32, 64\}$ and amount of dropout $\{0, 0.5\}$ hyperparameters [14].

To ensure the network trains until it convergences, the network is trained for a minimum of 50 epochs, after which the training is terminated when no improvements in validation accuracy is found in the last 20% of epochs. Early stopping is then implemented by selecting the epoch at which the model achieved its highest classification accuracy on the validation set.

IV. ANALYSIS AND RESULTS

A. Classification performance

The goal of this experiment is to analyse the behaviour of a CNN classifier on two modulation schemes of different types and orders that exposes fundamental characteristic when using a data point driven constellation diagram input.

From the classification accuracy and average class recall of the baseline architecture network in Figure 1, we can see that the model classifies well for SNRs above 0 dB. At lower SNRs the accuracy decreases as the signal falls below the noise floor. We also see an increase in accuracy when the number of kernels is increased to 36 kernels. Varying other hyperparameters revealed that adding dropout and using a weight decay value of 0.01 also increases accuracy and that the model generalises better on the validation set. The final hyperparameters for the baseline architecture can be found in Table II.

TABLE II
HYPERPARAMETERS OPTIMISED FOR THE BASELINE ARCHITECTURE.

Parameter	Value
Learning rate	0.0001
Batch size	32
Dropout	0.5
Weight decay	0.01
Kernel width	36

An interesting observation to make from Figure 1 is the unexpected improvement of the declining 8-PSK classification accuracy at very low SNR. The 16-QAM classification also increases slightly, but not as much as 8-PSK. This observation is strange, as we usually expect modulation classifiers to show reduced classification ability as noise increases until a reliable classification can no longer be made and the network shows 50% classification accuracy.

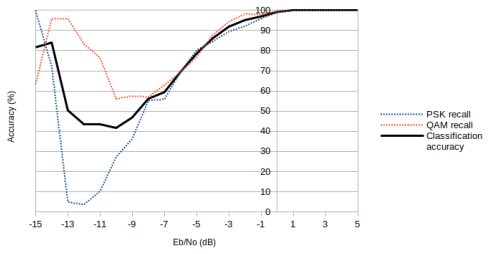


Fig. 1. Average classification accuracy (over 3 seeds) of the baseline architecture using optimised hyperparameters, for the evaluation data set with range of -15 dB to 5 dB. The average recall of 16-QAM and 8-PSK, respectively, is also shown.

B. Analysis of low SNR artefact

In order to find out why this increase in accuracy at lower E_b/N_o exists, we investigate whether this effect is due to the boundaries of the range of SNRs evaluated. Using the same training hyperparameters, the network was retrained for two ranges of the SNR, namely [-20;0] and [-10;10]. This investigation is also used to establish if the improvement in accuracy is tied to certain SNRs, or to a E_b/N_o position in the range. In addition, we investigate the effect of architectural changes to the neural network to ensure the artefact is not caused by lack of representational ability. The neural network will be adapted in the following ways:

- Increasing the dense layer node count from 100 to 1 000
- Changing the convolution layer kernels from 36 to 512
- Adding an extra convolution layer

To ensure well-defined results, each one of these changes is applied independently from the other.

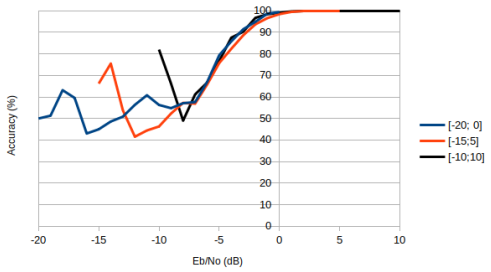


Fig. 2. Baseline network evaluation set performance when the network is trained on -20 dB to 0 dB, -15 dB to 5 dB and -10 dB to 10 dB SNR ranges, respectively.

When using the same baseline architecture as before but changing the E_b/N_o range, Figure 2 shows a similar trend to that observed before. Figure 2 indicates that the increase in accuracy occurs at lower E_b/N_o values, irrespective of the input data range. However, it is observed that the accuracy increase does not appear at a specific E_b/N_o. It should be noted that accuracy fluctuations only appear after the 0 dB accuracy descent and results near and above the noise floor increase gradually as expected.

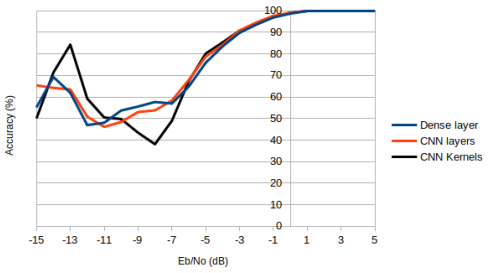


Fig. 3. Evaluation set performances when the dense layer of the baseline network is increased to 1 000 nodes ('Dense layer'), the baseline network's convolutional layers are increased from 2 to 3 ('CNN layers') and the baseline network's convolution kernels are increased to 512 ('CNN kernels').

When increasing the size and complexity of the network, Figure 3 shows a similar trend to that observed in the baseline architecture, except for variations in the average validation accuracy. Some of the methods change the shape and intensity of the accuracy increase in the E_b/N_o range, but all still exhibit the same artefact.

C. SNR-specific training

When observing Figure 1, we note that the increase in accuracy at lower E_b/N_o resembles models trained with SNR pairs selection [15]. With pairs selection, the network is only trained using two E_b/N_o data sets, instead of the entire range. In some instances this may cause an increase in accuracy surrounding the selected E_b/N_o pairs, especially in the low E_b/N_o range. To determine if this training method can give insight into the occurrence of the increase at low SNRs, we test how well the network can classify a single SNR's data by training baseline networks on only single SNR data sets. We also test the generalisation of each network over the entire SNR range to identify the classification abilities of our network on low SNR data.

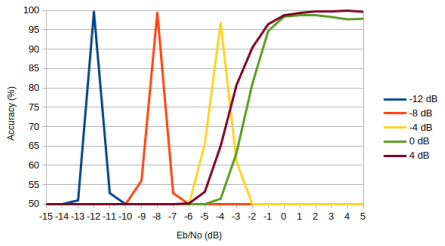


Fig. 4. The performance of five baseline networks, each trained on a specific dB value of SNR data and evaluated on the evaluation set.

From the classification accuracy of five baseline networks, in Figure 4, obtained from SNR-specific training, we see that each E_b/N_o could be classified above 90% accuracy over the -15dB to 5dB range, showing that the network can classify between the two modulation types, even when large amounts

of noise is added, if the task is restricted to a narrower noise range. Furthermore, this shows that the network is indeed able to classify accurately at lower E_b/N_o levels.

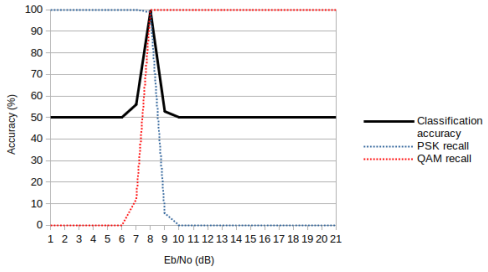


Fig. 5. Classification accuracy of a baseline network trained using only -8 dB data and evaluated on the evaluation set. The recall of 16-QAM and 8-PSK, respectively, is also shown.

From the classification accuracy and class recall of a baseline network trained on -8 dB data as shown in Figure 5, it is seen that networks trained on lower E_b/N_o data tend to generalise poorly to neighbouring E_b/N_o values and only show accuracy above 50% for one to two neighbouring E_b/N_o ranges before falling into a bias classification of 50%. Networks trained on E_b/N_ovalues above 0 dB E_b/N_o, however, show good generalisation, especially to higher E_b/N_o ranges than the E_b/N_o it is trained on. Accuracy of networks trained on high SNR data do however decrease when approaching and passing the noise floor at 0 dB E_b/N_o.

The knowledge that the network can accurately classify at any SNR level in our entire range, but then does not generalise well to other ranges, leads us to the observation that different classification criteria are being utilised for each SNR level, especially at lower SNRs. We can also see on which E_b/N_o level the network classifies accurately, not only by the increase in accuracy but also by the point where the network bias switches from 16-QAM to 8-PSK. This is also seen in our original network (Figure 1), where accuracy increases due to 16-QAM and 8-PSK classifications crossing over at lower E_b/N_o levels. These observations suggest the hypothesis that the network is prioritising certain E_b/N_o levels, or classification criteria, over others in the training process. By prioritising certain lower E_b/N_o levels the network increases the average validation accuracy.

D. Adding SNR as a feature

Knowing that the model can achieve an accuracy near 100% for any of the E_b/N_o levels within our range and the hypothesis that the network is selecting specific E_b/N_o ranges to optimise for, we turn to literature to find possible clarification of this behaviour. Several deep learning AMC networks [1] show increased accuracy when the E_b/N_o dB of the given constellation diagram is provided to the network to aid in classification. Providing the E_b/N_o level might allow the network to better optimise for the entire range, as it will not be blindly selecting an area in the E_b/N_o range to optimise for.

We provide the oracle E_b/N_o dB value to the network to test if this additional information aids the network in optimising better in the lower E_b/N_o range. The E_b/N_o value is provided to the linear layer of the network after the initial feature extraction has taken place in the convolutional layers, and prior to classification based on the extracted feature maps. By providing the E_b/N_o values, we test if the network will be able to adapt the classification criteria based on the E_b/N_o level.

From the classification accuracy of the network provided with SNR data in Figure 6, we observe a substantial increase in the validation accuracy across the entire E_b/N_o range. By providing the network with oracle E_b/N_o values, the network is able to adjust its classification criteria based on the amount of noise on the constellation data points.

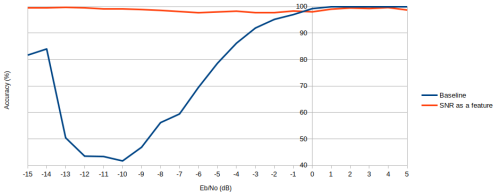


Fig. 6. Classification accuracy on the evaluation set, comparing the baseline network and a network where the SNR of the modulation scheme is presented to the network as a feature.

E. Effect of SNR estimation accuracy

By providing the oracle E_b/N_o data, we create a much-improved classifier for our binary classification problem. We do, however, know that the performance greatly relies on the provision of E_b/N_o values using SNR estimation at the receiver. This affects the implementation of this network in practical applications as noise level estimation accuracy tends to decrease at lower E_b/N_o levels [2]. To further understand the robustness and generalisation of the SNR-tagged network, we performed a sensitivity analysis.

The sensitivity analysis is conducted by generating statistical noise within a given range, as if an error was made when estimating the E_b/N_o value of the received signal, and adding that to the provided E_b/N_o data point when making a classification. The evaluation set and best performing baseline network is used for this analysis.

The results of the sensitivity analysis results for the baseline architecture is shown in Figure 7. We observe that variations within a 0.5 dB E_b/N_o range do not affect classification accuracy substantially, since all E_b/N_o still achieve above 95% accuracy. It is only after variation over 1 dB is introduced that the network's classification accuracy is noticeably reduced, especially at lower E_b/N_o levels.

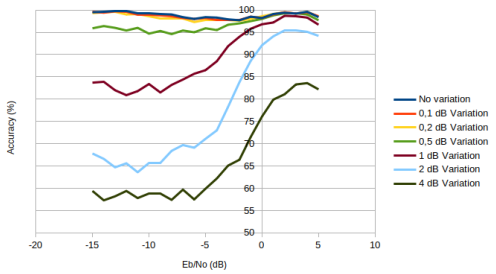


Fig. 7. A sensitivity analysis of SNR-tagged networks where the SNR input is corrupted with increased levels of stochastic variance. This mimics the effect of inaccurate SNR estimators.

This reduction in accuracy can be attributed to the network being trained with E_b/N_o step sizes of 1 dB, since as the provided SNR value moves closer to a neighbouring value, the generalisation of the classification criteria is reduced. The generalisation effect can also be observed at higher E_b/N_o levels as high accuracy levels are still achieved, even at high E_b/N_o variation. This observation once again highlights the specificity of the classification criteria needed to classify accurately at low E_b/N_o levels as opposed to above the noise floor.

V. Conclusion

This paper uncovers and investigates an artefact that occurs when implementing a modulation classifier for two modulation schemes, which provides constellation diagram data points as input to a CNN. It is found that the network can classify accurately at low SNR levels when only trained using specific E_b/N_o's data and that it generalises poorly to neighbouring E_b/N_o values. Knowing that the problem does not lie with the representational capacity of the network but rather with how the network models the task, the SNR values are provided as an additional input feature. This technique significantly increases accuracy at low E_b/N_o values as the network now has reference to the classification criteria to select when making a prediction. A sensitivity analysis of the effect when the additional input features are corrupted shows the weak generalisation of the classification criteria by highlighting the drop in performance when SNR estimation accuracy is low. This means that this network will only be of use if a SNR estimator that can accurately predict E_b/N_o at low SNR is used. Moving forward, this network and method could be further researched to improve AMC for low SNR environments.

Acknowledgement

The authors gratefully acknowledge the financial support of this study by the Telkom CoE at the NWU and Hensoldt South Africa.

References

[1] X. Xie, Y. Ni, S. Peng, and Y.-D. Yao, "Deep learning based automatic modulation classification for varying SNR environment," in *2019 28th Wireless and Optical Communications Conference (WOCC)*, 2019, pp. 1–5.

[2] W. Yongshi, G. Jie, L. Hao, L. Li, W. Zhigang, and W. Houjun, "CNN-based modulation classification in the complicated communication channel," in *2017 13th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*, 2017, pp. 512–516.

[3] J. L. Xu, W. Su, and M. Zhou, "Likelihood-ratio approaches to automatic modulation classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 4, pp. 455–469, 2011.

[4] J. P. Mouton, M. Ferreira, and A. S. Helberg, "A comparison of clustering algorithms for automatic modulation classification," *Expert Systems with Applications*, vol. 151, p. 113317, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420301421

[5] Y. Wei, S. Fang, and X. Wang, "Automatic modulation classification of digital communication signals using SVM based on hybrid features, cyclostationary, and information entropy," *Entropy*, vol. 21, no. 8, 2019. [Online]. Available: https://www.mdpi.com/1099-4300/21/8/745

[6] F. Meng, P. Chen, L. Wu, and X. Wang, "Automatic modulation classification: A deep learning enabled approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 760–10 772, 2018.

[7] S. A. Ghunaim, Q. Nasir, and M. A. Talib, "Deep learning techniques for automatic modulation classification: A systematic literature review," in *2020 14th International Conference on Innovations in Information Technology (IIT)*, 2020, pp. 108–113.

[8] S. Peng, H. Jiang, H. Wang, H. Alwageed, Y. Zhou, M. M. Sebdani, and Y.-D. Yao, "Modulation classification based on signal constellation diagrams and deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 718–727, 2019.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[10] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*, 2020, https://d2l.ai.

[11] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2019, preprint on webpage at https://arxiv.org/abs/1803.08375.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, preprint on webpage at https://arxiv.org/abs/1412.6980.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, preprint on webpage at https://arxiv.org/abs/1207.0580.

[15] S. Ramjee, S. Ju, D. Yang, X. Liu, A. E. Gamal, and Y. C. Eldar, "Fast deep learning for automatic modulation classification," 2019, preprint on webpage at https://arxiv.org/abs/1901.05850.

Andrew Oosthuizen received his B.Eng Computer and Electronic Engineering degree in 2020 and is currently a first year M.Eng student at the North-West University in South Africa. He is working as a dual member of the TeleNet research group in the Telkom CoE, and of MuST, a CAIR-affiliated research group at NWU specialising in deep learning.

A.4.2 Multi-Layer Perceptron for Channel State Information Estimation: Design Considerations

Multi-Layer Perceptron for Channel State Information Estimation: Design Considerations

Andrew Oosthuizen*†‡, Marelie H. Davel†‡§, Albert Helberg†
†*Faculty of Engineering, North-West University*, South Africa.
‡*Centre for Artificial Intelligence Research (CAIR), South Africa.*
§*National Institute for Theoretical and Computational Sciences (NITheCS), South Africa.*

aj.oosthuizen.ao@gmail.com
marelie.davel@nwu.ac.za
albert.helberg@nwu.ac.za

***Abstract*—The accurate estimation of channel state information (CSI) is an important aspect of wireless communications. In this paper, a multi-layer perceptron (MLP) is developed as a CSI estimator in long-term evolution (LTE) transmission conditions. The representation of the CSI data is investigated in conjunction with batch normalisation and the representational ability of MLPs. It is found that discontinuities in the representational feature space can cripple an MLP's ability to accurately predict CSI when noise is present. Different ways in which to mitigate this effect are analysed and a solution developed, initially in the context of channels that are only affected by additive white Guassian noise. The developed architecture is then applied to more complex channels with various delay profiles and Doppler spread. The performance of the proposed MLP is shown to be comparable with LTE minimum mean squared error (MMSE), and to outperform least square (LS) estimation over a range of channel conditions.**

***Index Terms*—Channel State Information, Deep learning, Multi-Layer Perceptron, Long-Term Evolution**

I. Introduction

Channel state information (CSI) is an integral part of wireless communication standards, and combines all channel impairments into a single estimation. Standards such as long-term evolution (LTE) and WiFi use pilot symbols, known to both the receiver and transmitter, to obtain CSI with one of several estimation methods that vary in computational cost. CSI is used to quantify the quality of the channel, which can then be used to make decisions on modulation scheme selection and transmission frequencies. CSI can also be used to equalise received transmissions by applying the inverse of the channel conditions, thus restoring data to its transmitted state if the CSI is of adequate accuracy [1], [2].

As deep learning methods have gained traction in CSI estimation in recent years [3], the field has seen many implementations outperform statistical methods such as least square (LS) [4], maximum likelihood (ML) [5] and minimum mean squared error (MMSE) [6] estimation. These deep learning implementations presented the input features to the CSI estimation networks in several different ways and it is unclear how the representation, specifically, affects the training procedure or performance of neural networks.

The paper is organised as follows: Related work is presented in Section II. Section III discusses the experimental setup, including the dataset setup, MLP architecture and training protocol; and is followed by an analysis of the results in Section IV. Within Section IV, the feature representation, hyperparameter choices and observed performance are discussed.

II. Background

Deep learning has been successfully applied to the CSI estimation problem using different deep learning architectures, such as recurrent neural networks (RNNs), long short-term memory (LSTM) networks and convolutional neural networks (CNNs) [3]. The attractiveness of deep learning methods has been accredited to the computational efficiency of deep learning models (once trained) over techniques such as MMSE that use resource-consuming matrix calculations. In addition, Jiang et al. [7] state that the non-linear capabilities of deep learning networks play an important role in estimating non-linear channel conditions more effectively than linear statistical methods. Finally, the data-driven nature of deep learning is fitting for telecommunication implementations, as ExaBytes of data is transmitted yearly [8] and real-world conditions can easily be simulated to generate data.

Due to the data-driven nature of deep neural networks (DNNs) [9] and the ease of training a multi-layer perceptron (MLP) compared to other deep learning architectures, it is a natural choice to apply an MLP to this task. MLPs and deep neural networks have been shown to outperform LS methods and contend with MMSE methods under complex channel conditions [10], [11]. The task can be framed in different ways, with either pilot points data or an already generated CSI estimate of low quality as input. In related work, it was seen that this data can be presented in several different ways to DNNs, with main representations: as the angle and absolute values [12], as complex values directly [13], and as quadrature and in-phase components (QI) of this complex data [10].

More recently deep learning architectures have been implemented to reduce the overhead caused by CSI in resource allocation problems [14] and for estimation of CSI feedback for 5G implementations [15]. These two papers both utilise

different architectures and input features with Godala et al. [15] using CNNs with real and imaginary input features and Khan et al. [14] using DNNs and angular input features.

DNNs have many hyperparameters that play a role in their performance. Some of these hyperparameters are related to the dataset, such as the presentation of features or sample size, while others are related to the architecture of the DNN. There are also hyperparameters that control how the networks are trained, such as optimisers and learning rates. When confronted with these many interacting variables, it is crucial to understand the impact of choosing one hyperparameter over the other. Due consideration should also be given to the specific range of hyperparameter values used when applying hyperparameter searches, as some hyperparameter values perform better in combinations with specific values of other hyperparameters.

This paper investigates the performance of an MLP in predicting the CSI of a single input, single output (SISO) channel setup under various LTE [16] channel conditions. Within this context, the following questions are addressed:

- How should features be represented for CSI estimation?
- How sensitive are results to MLP architectural choices?
- What order of performance can be expected for channels with different complexities?

In answering these questions, several observations are made with regard to which feature representations perform best, and how architectural choices influence the performance of MLPs for CSI estimation.

III. Experimental setup

Synthetic data is generated and used to train an MLP. The data generation process is described in Section III-A, the MLP architecture (including the varied elements) in Section III-B, and the hyperparameter optimization process in Section III-C.

A. Dataset setup

The initial dataset is generated using a SISO setup with no noise and no delay profile through the LTE toolbox available in Matlab®. The downlink transmission used in this study makes use of orthogonal frequency-division multiplexing (OFDM) with a bandwidth of 10MHz and short cyclic prefixes. To ensure that only the performance of the MLP is measured, no link level techniques are used and control frames are removed as well.

Each data sample consists of a single slot containing 4 pilot symbols. In Figure 1 we see that slots contain 7 OFDM signals with 12 sub-carriers, each depicting a Quadrature Phase Shift Keying (QPSK) modulated symbol. The input data is provided to the MLP from the four pilot symbols, extracting 2 features per symbol and then flattening the data into a single tensor with a width of 8. This process is also applied to the target data using the true CSI for the entire slot to produce a 168-wide tensor. It is thus expected for the MLP to estimate the true 168-dimensional CSI given the 8-dimensional input.

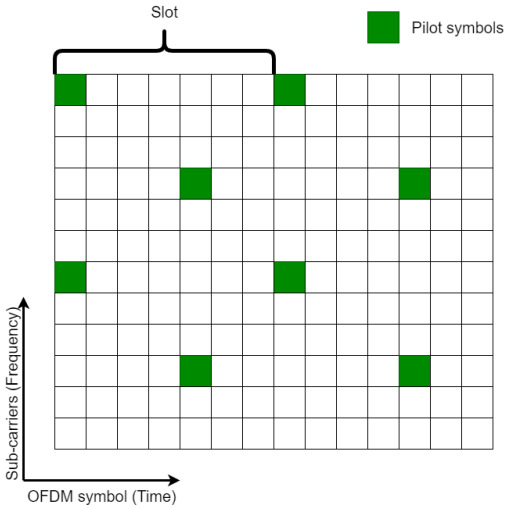


Fig. 1. LTE frames contain pilot symbols used for CSI estimation.

Using these principles, several datasets are created to train MLPs. The least complex dataset contains only primary multipath effects since no delay profile or noise is added to the channel. This is referred to as the 'noiseless' dataset. The 'noisy' dataset is similar to the noiseless dataset, except that it adds 20dB additive white Gaussian noise (AWGN) to the channel. To further test the MLP's ability, delay profiles are added to the noisy dataset making the channel more complex. Each delay profile incrementally increases the complexity of the multipath components sent over the channel. The Extended Pedestrian A model (EPA), Extended Vehicular A model (EVA) and Extended Typical Urban model (ETU) [17] respectively represent low, medium and high delay spread environments. These three datasets are called the 'delay profile' datasets. Lastly, a 'Doppler fading' dataset is constructed by adding Doppler fading to the EVA dataset. This process of increasing complexity is used to investigate the effect of different channel conditions on the MLP's performance.

Each dataset consists of a training set of 20 000 samples and a validation set of 5 000 samples. An unseen test for each dataset consisting of 5 000 samples is also generated. For each test set, the LS and LTE MMSE estimates are calculated as a comparative baseline. The LS estimate has low complexity and is obtained by linearly extrapolating the CSI from known pilot data, while the standardised LTE MMSE downlink estimation is obtained as described in [18]. These two methods give an indication of how well the MLP is performing against statistical CSI estimation methods.

B. MLP architecture

The deep neural network used in this paper is an MLP, as MLPs consist of fewer architectural hyperparameters than other neural networks such as CNNs or LTSMs. More com-

plex architectures add an additional architecture selection process during training. This paper thus uses the simplest network to probe feature representation which other networks can further utilise.

The MLP network has a general architecture consisting of several layers of fully-connected neurons, each containing the same set amount of neurons per layer. Between each layer lies an activation function, as well as a batch normalisation layer [19]. The activation function introduces non-linearity into the model, enabling it to learn non-linear data distributions and perform complex regression tasks. TanH is a commonly used activation function for CSI prediction and has shown promising results over other activation functions for this task [20]. Batch normalisation is a concept that was developed by Ioffe et al. [19] originally "to reduce internal covariate shift" within neural networks but with additional benefits becoming apparent over time. This method enables accelerated training in neural networks by normalising activations between layers. Ioffe et al. also claim that less careful initialisation of networks is needed to obtain a successfully trained network if batch normalisation is present.

The depth (the number of hidden layers in a network) and width (the number of nodes in a hidden layer) of the network are considered structural hyperparameters in Section III-C. Structural hyperparameters control the size of a network and thus its representational capability. The representational capacity of a DNN is a measure of the complexity of the data distribution that the network is able to learn. It is important to make the network large enough to solve the problem, however, making the network too large makes it computationally expensive to train and has been known to create vanishing gradient problems.

C. Training protocol

All networks are trained using the Adam [21] optimiser and the mean square error (MSE) loss metric. Adam is selected for its ability to adapt the learning rate of the training algorithm for each parameter individually. MSE is used as loss function since this research is interested in the difference between the real CSI and estimated CSI. All networks are trained to convergence, and early stopping is used: the model is selected at the epoch of best validation accuracy.

Different hyperparameters are assessed by using a hyperparameter grid search in which all possible combinations of the selected hyperparameter values are applied. Each combination of hyperparameters is also trained over three random initialisation seeds to ensure that strong or weak initialisation does not affect trend analysis over the sweep. Presented loss and bit error rate (BER) results are the averages over the three random seeds of a specified hyperparameter set. Since exhaustive hyperparameter searches are conducted it is important to select a wide range of hyperparameter values that can be narrowed down if needed.

In the initial grid search, the protocol exhaustively search over possible combinations of depth {1, 2, 3, 4} and width {10, 100, 1 000}. At the same time the protocol also sweeps over batch size {16, 32, 64, 128} and learning rate{1e-02, 1e-03, 1e-04, 1e-05}, referred to as the training hyperparameters. This grid search thus contains 576 possible hyperparameter combinations, including initialisation seeds. These hyperparameter ranges are chosen as the structural hyperparameters provide sufficient representational capacity while still being computationally feasible to train. The learning rate is selected over a wide range of values, while the batch size selection remains large enough to see effect but still be memory conscious.

The best performing networks are chosen based on the lowest validation loss and are tested on the unseen test set. This protocol ensures that networks do not overfit training data and can generalise to unseen data.

IV. Analysis and results

A. Feature representation

Following Erdogmus et al. [12], the angle and absolute values are initially used as input features. The initial model is constructed using the noiseless dataset to train the MLP; this is done to create a baseline for comparison and to validate the training process. Note that, when a BER of 0 is calculated on a specific dataset, it is reported as δ. For the noisless dataset it is expected that the MLP obtains a BER of δ, as this is what LS obtains on these channel conditions.

When evaluating these networks on the test set, a loss of 4.61e-2 and BER of 8.33e-4 are obtained. These results are concerning as a simple LS implementation can obtain a BER of δ. When further analysing the results of the best performing MLPs, it is found that the network has difficulty estimating the CSI angle close to the discontinuity within the angle representation of the data. This discontinuity is found at the point where the CSI rotates past 3.14 (PI) or 180 degrees. Angles use a radian representation where the angle abruptly moves from 3.14 to -3.14 instead of continuing linearly to 3.15 when passing 180 degrees from the original starting point. Examining the angle component of OFDM symbols in Table I, it is observed that the network has difficulty correctly predicting the angle of the CSI when approaching PI. The incorrectly estimated sub-carriers are shown in bold in Table I.

TABLE I
PREDICTED CSI ANGLE OVER TWO OFDM SIGNALS COMPARED TO THE TARGET CSI ANGLE OVER THE SAME OFDM SIGNALS

	Predicted CSI (OFDM symbol)		Target CSI (OFDM symbol)	
Sub-carrier	1	2	1	2
1	**1.36**	**1.35**	3.09	3.09
2	3.05	3.05	3.05	3.05
3	3.00	3.01	3.01	3.01
4	2.88	2.89	2.96	2.96
5	2.85	2.85	2.92	2.92
6	2.80	2.8	2.88	2.88
7	2.77	2.77	2.83	2.83
8	2.72	2.72	2.79	2.79
9	2.67	2.68	2.75	2.75
10	2.56	2.55	2.71	2.71
11	2.52	2.51	2.66	2.66
12	2.46	2.45	2.62	2.62

After removing batch normalisation from the MLP (reasoning as discussed in Section IV-B), a BER of δ is obtained on the noiseless data using the angle and absolute value data representation.

When experimenting with the same feature representation and the noisy dataset, another concerning behaviour is observed: As a baseline, the best performing network trained on noiseless data achieves a loss of 2.31e-1 and BER of 5.16e-3 on the noisy test set. After following the training protocol, the best performing networks (as measured using the validation set) are obtained. When applying these best performing networks to the test set, a loss of 1.28e-1 and BER of 1.72e-2 are observed. This behaviour characterises a dysfunctional training process as the loss is not correlated to the BER: it is expected that, when the predicted CSI looks more similar to the real CSI, shown as a lower MSE loss, the performance, shown as BER, would improve.

Examining the predicted CSI, it can be observed that adding noise to the channel again introduces the discontinuity problem, similar to the results in Table I. This is not unexpected, as the network now struggles to find the precise point at which the discontinuity exists due to the added noise. Ahmed et al. [22] explain how deep learning methods may struggle using data with non-euclidean features and perform much better on data presented in euclidean space. Therefore, it is proposed to change the extracted features from absolute value and angle to the QI of the CSI representation. This places the data representation on a continuous euclidean plane that may be interpreted better by the MLP.

The QI representation is implemented on the noisy dataset using the hyperparameter sweep protocol. Analysing the results, a decrease in the loss of the training hyperparameters are observed. Testing the best performing networks a loss of 5.23e-3 and BER of δ is obtained. These results show that changing the features used by the MLP has increased the MLP's ability to generate accurate CSI. It is assumed that this is the result of using a feature space of euclidean nature, which is more fitting to the MSE loss function.

B. Effect of hyperparameter choices

In this section, the effect of structural hyperparameters, training hyperparameters and the effect of batch normalisation in the training process is discussed.

In examining the structural hyperparameters when training the MLP on the noiseless data set using the angle and absolute representation, it was found that MLPs with deeper and wider layers perform better than smaller networks. These results (not shown here) indicate that increasing the representational ability of the MLP by providing a larger architecture decreases the loss on the validation set, thus lowering the BER. For the remainder of the experiments, moving on from the noiseless dataset, a single MLP architecture which consists of 4 layers of a 1 000 nodes each is used, to ensure consistent results and to simplify the training analysis. This decision can be made as it has been confirmed that the network has the representational ability to make CSI predictions.

The analysis in this paper always considers training hyperparameters in the hyperparameter sweeps. The initial sweeps (not shown here) on noiseless data, with an angle and absolute representation, showed that a learning rate of 1e-3 performed best over a range of runs. It was also found that performance increases as batch sizes become larger. This leads to the conclusion that increasing the amount of data processed before a gradient update (the batch size), improves results. Additional hyperparameter searches are thus done over the same learning rate range and an extended batch size range: {32, 128, 256, 512}. The batch size is extended as previous results indicate improved learning environments at higher batch sizes.

As stated in Section IV-A the results of the angle and absolute data representation on noiseless data are less than optimal, especially when compared to LS results, due to the presence of a discontinuity in the data representation. To help the MLP learn how to interpret the discontinuity found in the representational plane the decision to change hyperparameters is made. The activation function and batch normalisation layers can be altered when inspecting structural hyperparameters. There are no grounds for changing the TanH activation function as it has been widely used in the CSI field. Thus the batch normalisation layer is examined.

Santurkar et al. [23] suggest that batch normalisation does not succeed in reducing internal covariance shift but rather in smoothing the optimisation landscape that leads to more predictive gradient behaviour. This is done by reparameterising the underlying optimisation problem. Furthermore, smoothing the optimisation landscape may be why the MLP is having trouble identifying the discontinuity, as the exact position of where the MLP acknowledges the discontinuity may be a sharp point in the optimisation landscape that is smoothed over.

This hypothesis is tested by running a hyperparameter sweep on the noiseless dataset but removing batch normalisation layers from the MLP architecture. Table II depicts the architectural hyperparameters for a specific set of learning hyperparameters from the hyperparameter sweep. When examining the results, it is observed that the best performing networks have loss results that are better by several orders of magnitude, when compared with the batch normalisation sweeps. Applying this network on the test set, a BER of δ is obtained, which is the expected result from an MLP on this dataset.

TABLE II
BEST PERFORMING STRUCTURAL HYPERPARAMETERS TEST LOSS RESULTS FOR MLP TRAINED ON NOISELESS DATA AND NO BATCH NORMALISATION LAYERS

	Depth			
Width	1	2	3	4
10	8.41e-2	9.45e-2	1.27e-1	1.95e-1
100	5.41e-2	4.46e-2	2.54e-2	1.14e-3
1 000	4.71e-2	1.65e-3	9.49e-8	6.33e-8

From these results, it can be observed that the batch normalisation layer smooths the results over different hyperparameter combinations, further referred to as the hyperparameter landscape. This can be seen from the more diverse sweep

results that are obtained when removing batch normalisation. However, it is speculated that this smoothing prevented the training protocol from finding a point in the hyperparameter landscape where the discontinuity is recognised.

The hypothesis is further investigated by applying the QI data representation and implementing a hyperparameter sweep on noisy data using networks with and without batch normalisation. If the assumptions thus far are correct, there should be no reason for batch normalisation to decrease the network's performance significantly. These results are reported on over different batch sizes and learning rates with and without batch normalisation in Table III and Table IV, respectively.

TABLE III
TEST LOSS RESULTS OVER GROUPED TRAINING PARAMETER FOR NOISY DATA WITH QI VALUES AS FEATURES

	Learning rate			
Batch size	1e-5	1e-4	1e-3	1e-2
32	5.28e-3	5.22e-3	5.49e-3	2.64e-2
128	5.32e-3	5.22e-3	5.49e-3	1.40e-2
256	5.33e-3	5.26e-3	5.56e-3	9.37e-3
512	5.36e-3	5.31e-3	5.56e-3	7.59e-3

TABLE IV
THE SAME SETUP AS IN TABLE III, EXCEPT WITH BATCH NORMALISATION

	Learning rate			
Batch size	1e-5	1e-4	1e-3	1e-2
32	5.47e-3	5.64e-3	5.37e-3	6.33e-3
128	5.47e-3	5.5e-3	5.36e-3	5.75e-3
256	5.48e-3	5.48e-3	5.47e-3	5.66e-3
512	5.46e-3	5.47e-3	5.57e-3	5.59e-3

When Analysing these tables, a smoother set of results in the same range as the MLP using a QI feature set and no batch normalisation is noted. From the results found in IV, batch normalisation is reintroduced into the MLP architecture as a smoother hyperparameter landscape is obtained, ensuring networks that perform in the expected range from the training process. In the results presented in Table III and Table IV it is seen that this is done at a negligible loss in accuracy but would provide a speedup in training [19].

Thus, batch normalisation is suspected of having a smoothing effect in the optimisation space. While this feature is generally desired in the optimisation process, it can potentially be damaging when implementing a non-euclidean data representation, which has particular optimisation points.

C. Observed performance

To test the ability of the MLP CSI estimator, the three delay profile datasets are applied. The networks are trained making use of the same training protocol as described in Section III-C except for the following, as motivated in Section IV-B:

- The structural hyperparameters are fixed at 4 layers and 1 000 neurons.
- Adapting the range of batches searched over to {32, 128, 256, 512}.

In Table V the BER results of the best performing network of each delay profile tested on all delay profiles (the corresponding delay profile as well as others) are displayed. It is observed that each network performs best on the data it was trained on, but does not fail when applied to other delay profiles, showing a degree of cross-condition generalisation. Comparing the results to an LS implementation it is noted that the network improves on the LS BER, especially on EPA and EVA data.

TABLE V
BER OF MLP NETWORKS TRAINED ON THE DELAY PROFILE DATASETS AND APPLIED TO THE SAME OR OTHER DELAY PROFILE TEST SETS

Trained on	EPA	EVA	ETU
EPA	3.97e-3	1.10e-2	3.63e-2
EVA	4.82e-3	6.88e-3	1.57e-2
ETU	6.69e-3	8.35e-3	1.16e-2
LS Method	1.20e-2	1.43e-2	3.28e-2

The channel is made more complex by applying the Doppler dataset. Following the training protocol described previously, the results in Table VI are obtained. This table shows that the networks outperform LS and can compete with MMSE at lower Doppler frequencies, even with minimal statistical information on the channel conditions. Finally, it is observed that the MMSE method obtains similar BER performance for all Doppler frequencies.

TABLE VI
BER OF DIFFERENT EQUALISATION METHODS ON THE DOPPLER DATASET WITH VARIOUS AMOUNTS OF DOPPLER SHIFT

Doppler (Max Hz)	MLP	LS	MMSE
50	6.99e-3	1.38e-2	8.6e-3
100	8.01e-3	1.45e-2	8.6e-3
200	9.13e-3	1.5e-2	8.6e-3

V. CONCLUSION

This paper implements an MLP for CSI estimation over several OFDM symbols under various LTE channel conditions for a SISO system. Through an extensive model development process, an MLP architecture was found capable of contending with MMSE methods and improving LS estimation results in complex LTE channel conditions. These results can however only be obtained when utilising the correct hyperparameters for selecting and training the MLP network. First, using wide hyperparameter sweeps it is found that a discontinuous feature representation can obtain results comparable to LS in less complex channel conditions at the cost of removing batch normalisation. However, when considering the effect the feature representation has on the results, this research motivates for the importance of using a continuous euclidean feature representation when using complex channel conditions.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the financial support of this study by the Telkom CoE at the NWU and Hensoldt South Africa.

REFERENCES

[1] D. F. Carrera, C. Vargas-Rosales, N. M. Yungaicela-Naula, and L. Azpilicueta, "Comparative study of artificial neural network based channel equalization methods for mmWave communications," *IEEE Access*, vol. 9, pp. 41 678–41 687, 2021.

[2] Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *MIMO-OFDM wireless communications with MATLAB*. IEEE Press, 2011.

[3] C. Luo, J. Ji, Q. Wang, X. Chen, and P. Li, "Channel state information prediction for 5G wireless communications: A deep learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 227–236, 2018.

[4] E. Karami, "Tracking performance of least squares MIMO channel estimation algorithm," *IEEE Transactions on Communications*, vol. 55, no. 11, pp. 2201–2209, 2007.

[5] Z. Du, X. Song, J. Cheng, and N. C. Beaulieu, "Maximum likelihood based channel estimation for macrocellular OFDM uplinks in dispersive time-varying channels," *IEEE Transactions on Wireless Communications*, vol. 10, no. 1, pp. 176–187, 2010.

[6] J. Ma, S. Zhang, H. Li, N. Zhao, and A. Nallanathan, "Iterative LMMSE individual channel estimation over relay networks with multiple antennas," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 423–435, 2017.

[7] Z. Jiang, S. Chen, A. F. Molisch, R. Vannithamby, S. Zhou, and Z. Niu, "Exploiting wireless channel state information structures beyond linear correlations: A deep learning approach," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 28–34, 2019.

[8] Ericsson, Ericsson mobility report, 2017, available: https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf.

[9] W. Jiang and H. D. Schotten, "Deep learning for fading channel prediction," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 320–332, 2020.

[10] Y. Yang, F. Gao, X. Ma, and S. Zhang, "Deep learning-based channel estimation for doubly selective fading channels," *IEEE Access*, vol. 7, pp. 36 579–36 589, 2019.

[11] E. Balevi and J. G. Andrews, "Deep learning-based channel estimation for high-dimensional signals," *arXiv preprint arXiv:1904.09346*, 2019.

[12] D. Erdogmus, D. Rende, J. C. Principe, and T. F. Wong, "Nonlinear channel equalization using multilayer perceptrons with information-theoretic criterion," in *Neural Networks for Signal Processing XI: Proceedings of the 2001 IEEE Signal Processing Society Workshop (IEEE Cat. No. 01TH8584)*. IEEE, 2001, pp. 443–451.

[13] Y. Zhang, J. Wang, J. Sun, B. Adebisi, H. Gacanin, G. Gui, and F. Adachi, "CV-3DCNN: Complex-valued deep learning for CSI prediction in FDD massive MIMO systems," *IEEE Wireless Communications Letters*, vol. 10, no. 2, pp. 266–270, 2020.

[14] H. Khan, M. M. Butt, S. Samarakoon, P. Sehier, and M. Bennis, "Deep learning assisted CSI estimation for joint URLLC and eMBB resource allocation," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2020, pp. 1–6.

[15] A. R. Godala, S. Kadambar, A. K. R. Chavva, and V. S. Tijoriwala, "A deep learning based approach for 5G NR CSI estimation," in *2020 IEEE 3rd 5G World Forum (5GWF)*. IEEE, 2020, pp. 59–62.

[16] J. Zyren and W. McCoy, "Overview of the 3GPP long term evolution physical layer," *Freescale Semiconductor, Inc., white paper*, vol. 7, pp. 2–22, 2007.

[17] "E-UTRA, base station (BS) radio transmission and reception," 3GPP, May 2008.

[18] "Evolved universal terrestrial radio access E-UTRA; base station BS conformance testing," 3GPP TS 36.141, July 2018.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[20] H.-C. Tsai, C.-J. Chiu, P.-H. Tseng, and K.-T. Feng, "Refined autoencoder-based CSI hidden feature extraction for indoor spot localization," in *2018 IEEE 88th vehicular technology conference (VTC-Fall)*. IEEE, 2018, pp. 1–5.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, preprint on webpage at https://arxiv.org/abs/1412.6980.

[22] E. Ahmed, A. Saint, A. E. R. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten, "A survey on deep learning advances on different 3D data representations," *arXiv preprint arXiv:1808.01462*, 2018.

[23] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" *Advances in neural information processing systems*, vol. 31, 2018.

Andrew Oosthuizen received his B.Eng Computer and Electronic Engineering degree in 2020 and is currently a second year M.Eng student at the North-West University in South Africa. He is working as a dual member of the TeleNet research group in the Telkom CoE, and of MUST, a CAIR-affiliated research group at NWU specialising in deep learning.

A.4.3 Adversarial Training for Channel State Information Estimation in LTE Multi-Antenna Systems

Adversarial Training for Channel State Information Estimation in LTE Multi-Antenna Systems

AJ Oosthuizen[1,2][0000−0001−7471−6384], ASJ Helberg[1][0000−0001−6833−5163], and MH Davel[1,2,3][0000−0003−3103−5858]

[1] Faculty of Engineering, North-West University, South Africa
aj.oosthuizen.ao@gmail.com
albert.helberg@nwu.ac.za
marelie.davel@nwu.ac.za

Abstract. Deep neural networks can be utilised for channel state information (CSI) estimation in wireless communications. We aim to decrease the bit error rate of such networks without increasing their complexity, since the wireless environment requires solutions with high performance while constraining implementation cost. For this reason, we investigate the use of adversarial training, which has been successfully applied to image super-resolution tasks that share similarities with CSI estimation tasks. CSI estimators are usually trained in a Single-In Single-Out (SISO) configuration to estimate the channel between two specific antennas and then applied to multi-antenna configurations. We show that the performance of neural networks in the SISO training environment is not necessarily indicative of their performance in multi-antenna systems. The analysis shows that adversarial training does not provide advantages in the SISO environment, however, adversarially trained models can outperform non-adversarially trained models when applying antenna diversity to Long-Term Evolution systems. The use of a feature extractor network is also investigated in this study and is found to have the potential to enhance the performance of Multiple-In Multiple-Out antenna configurations at higher SNRs. This study emphasises the importance of testing neural networks in the context of use while also showing possible advantages of adversarial training in multi-antenna systems without necessarily increasing network complexity.

Keywords: Channel state information · Deep learning · Adversarial training · Multiple-In Multiple-Out systems · Long-Term Evolution

[2] Centre for Artificial Intelligence Research (CAIR), South Africa.

[3] National Institute for Theoretical and Computational Sciences (NITheCS), South Africa.

1 Introduction

Channel state information (CSI) estimation is a wireless communication technique used to estimate channel conditions and equalise channel impairments experienced by received data. Within the Long-Term Evolution (LTE) standard, CSI estimation uses pilot signals known to both transmitter and receiver to deliver a single estimation describing channel impairments [1]. In recent years, statistical CSI estimation and deep learning methods have obtained satisfactory results on this task [2]. However, less than optimal linear methods, such as least squares (LS) [3] and linear minimum mean squared error (MMSE) [4], are frequently applied due to the computational expense of more complex CSI estimation methods, making them infeasible to implement on user hardware.

For this reason, we investigate the performance of adversarially trained models since, while their training times are much higher than their non-adversarially trained counterparts, their computational expense remains the same when implemented. Recently Zhao et al. [5] implemented a generative adversarial network (GAN) architecture for CSI estimation and reported improved results due to the addition of adversarial training. We continue the work of Zhao et al. by investigating the effects of adversarial training on multi-antenna (MA) implementations. MA implementations are investigated since using antenna diversity techniques has become the standard in modern-day wireless communication and adds additional resilience to the impairments caused by fading channels [1, 6].

The main contributions of this paper are:

- Investigating the difference in performance between adversarial and non-adversarial training for CSI estimation tasks.
- Providing results that show how antenna diversity techniques obtain unexpected performance gains using different adversarial training methods.
- As a side note, we also discuss the importance of model selection in the context of use rather than solely on training environment performance.

The rest of the paper is organised as follows: Section 2 provides background on the main techniques referred to, before closely related work is discussed in Section 3. The experimental setup is described in Section 4. This setup is used for all experiments conducted and reported on in Section 5. Results and conclusions are summarised in Section 6.

2 Background

We briefly discuss the CSI estimation task,the super-resolution GAN (SRGAN) architecture and antenna diversity techniques in the context in which these are utilised.

2.1 Channel State Information

For the current analysis, CSI estimation is implemented using the pilot-based method described in the LTE standard [1]. Pilot signals are placed in specific

sub-carriers of orthogonal frequency division multiplexing (OFDM) modulated signals and are data points known to both transmitter and receiver. OFDM signals are two-dimensional transmissions that transmit data in the frequency and time domain and are thus represented as grids with frequency as the first dimension and time as the second. These signals are used according to Equation 1 to obtain the initial CSI H from the transmit antenna Tx and receive antenna Rx, where the sub-carrier k represents the position of the pilot signals:

$$H[k] = \frac{Rx[k]}{Tx[k]} \tag{1}$$

Using the initial CSI, we obtain an LS estimation by extrapolating from the known CSI pilot signals to obtain the LS CSI estimation $\hat{H}_{LS}$ [7, 6, 3]. This method is the simplest form of channel estimation and can be used to equalise received data, resulting in the equalised received data $\hat{Tx}$ according to Equation 2.

$$\hat{Tx} = \frac{Rx}{\hat{H}} \tag{2}$$

The process of equalisation is illustrated in Figure 1 where received data is being equalised using complete knowledge of the CSI, i.e. perfect CSI. Typically, the equalised data does not return to its original transmitted state since some of the receiver antenna noise cannot be estimated but only reduced using denoising methods.

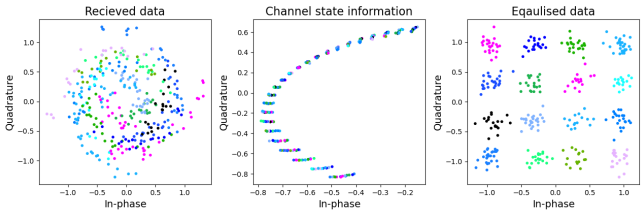


Fig. 1. An illustration of the process of equalising received data with 20dB antenna noise (left), by using perfect CSI (middle) to obtain equalised data (right) that can be demodulated without error.

Applying LS to the pilot signals to estimate the CSI has a low computational cost but does not provide state-of-the-art performance. Alternatively, methods such as MMSE provide improved performance at the cost of computational efficiency as second-order statistics of the channel need to be calculated and implemented using costly matrix multiplication methods.

Many wireless devices, such as internet of things devices, require real-time computation to ensure no delays are caused in time-sensitive applications. Soltani

et al. [8] compare a neural network (NN) CSI estimator to a classic CSI estimation method, finding that the NN outperforms the classic method. However, the NN requires 50 times the floating-point operations of the classic method to achieve this result. This encourages us to find ways of improving NN performance without increasing complexity.

2.2 Super Resolution GAN

Adversarial training is a training method in which a second neural network is introduced in the training process to compete with the primary or generator network. The second neural network, commonly called the discriminator, attempts to identify generated samples from ground truth samples. These networks are trained simultaneously; thus, one network's ability affects the training of the other. Adversarial training is implemented to train generator networks capable of generating samples closer to the ground truth than non-adversarially trained networks [9].

SRGAN [10] is a GAN designed for image super-resolution (SR) tasks, specifically image upsampling tasks. The adversarial training used by Ledig et al. for SRGAN is reported to push the generated SR images closer to the original images' manifold. This SR implementation results in a higher signal-to-noise ratio (SNR) and sharper high-frequency feature generation. SRGAN achieves these results by moving away from using the mean square error (MSE) to compare individual pixels to each other. Instead, SRGAN compares feature maps of the images obtained by a visual geometry group (VGG) network [11] in the adversarial training stage. Applying this VGG loss $l_{content}$ to images generated, the euclidean distance is measured between the feature maps of real and generated images. The loss function, described in Equation 3, averages the MSE loss between the feature maps, generated by the VGG-19 network ϕ, of the high-resolution image I^{HR} and the image generated from low resolution using the generator $G(I^{LR})$:

$$l_{content} = \frac{1}{ij}\sum_{y=1}^{i}\sum_{x=1}^{j}(\phi(I^{HR})_{x,y} - \phi(G(I^{LR}))_{x,y})^2 \tag{3}$$

Here i and j represent the respective dimensions of the feature maps. The content loss forms part of the final loss function l_{SRGAN} in Equation 4 by being added to the adversarial loss used to evaluate the generator using the discriminator $l_{adversarial}$:

$$l_{SRGAN} = l_{content} + 1e^{-3} * l_{adversarial} \tag{4}$$

SRGAN's generator consists of a convolution layer using a parametric rectified linear unit (PReLU) [12] activation function followed by several residual blocks and another convolution layer, all using batch normalisation [13]. Two 2× upsampler layers are added to upscale the physical dimensions of the feature maps before feeding the maps to an output convolution layer. All residual

blocks use 64 channels before these are increased to 256 channels in the upscaling blocks.

The discriminator's structure closely follows a VGG network with eight convolution layers. Each of these layers reduces the feature maps' size while increasing the number of channels until the output is passed to a 1 024 node dense layer followed by a single node layer connected to a sigmoid activation function. This sigmoid output is then used with binary cross-entropy (BCE) loss to determine whether samples are real or generated.

2.3 Diversity Techniques

The principle applied in antenna diversity is to suppress the effects of fading channels by creating spatial diversity between antennas. The idea behind antenna diversity is that the probability that multiple independent channels are experiencing aggressive fading conditions at the same time is unlikely. We call this utilisation of different channels 'diversity gain', which is used to improve the bit error rate (BER) performance of MA systems.

In this work, we focus on both receiver and transmitter antenna diversity methods. For receiver diversity, we implement equal gain combining (EGC) [6], where a single transmit antenna transmits to multiple receiver antennas. Each antenna calculates the CSI between itself and the transmit antenna and equalises the received data. These streams of equalised data are then combined by averaging each corresponding sub-frame over the other receiver antennas.

To implement transmitter diversity, we utilise multiple transmit antennas with single or multiple receive antennas. We use orthogonal space-time block codes (OSTBC), specifically an Alamouti encoder [6] to enable communication in this MA environment. Alamouti encoders utilise the multiple transmitter antennas available by creating space-time block codewords with complex orthogonality. The Alamouti encoder deconstructs a single data stream into multiple data streams that, when transmitted, create a complex orthogonal matrix of the original symbols over the channel. Alamouti encoders require accurate CSI of all possible channel combinations to provide accurately equalised data. Link-level implementations use these MA methods to convert time-varying channels to additive white Gaussian noise (AWGN)-like channels.

3 Related Work

The CSI estimation problem has been approached using several deep learning architectures such as recurrent neural networks, long short-term memory networks and convolutional neural networks (CNNs) [14]. Specifically, CNN architectures have regularly appeared in CSI estimation papers due to the image-like nature of OFDM grids and the fading characteristics of many wireless channels. These CNN implementations outperform LS and are competitive with methods such as MMSE without calculating second-order channel conditions [15, 16]. The

computational cost of methods is an important consideration in wireless communications as methods with lower computational cost and adequate performance are regularly selected over better-performing methods. In the LTE standard this is the case as an adapted LS method is used for CSI estimation over methods such as MMSE [1].

In the image processing field, CNNs have obtained state-of-the-art results in denoising and upsampling tasks when paired with adversarial training methods [10]. In the telecommunications domain, we have also seen increased use of GANs, primarily to model wireless channels. By modelling channels with GANs, autoencoders and other end-to-end neural networks can pass gradients through the propagation channel model, simplifying the training and design process of end-to-end implementations [17].

Zhao et al. [5] applies GANs to the CSI estimation problem, hypothesising that the GAN architecture generates CSI predictions closer to the target samples' manifold than traditional methods. Zhao et al. use an adaptation of SRGAN to generate high-resolution CSI from pilot signals. However, several changes are made to the original SRGAN architecture to provide improved results for the CSI estimation task. As illustrated in Figure 2, the upsampling layers are removed, and a pre-upsampling methodology is implemented. The pre-upsampling is applied by using pilot signals to construct the LS CSI estimate, which would represent the low-quality input in the original SRGAN. By doing this, the upsampling of the pilot signals is essentially removed from the neural network. The authors show improved results when implementing this pre-upsampling method. It is, however, unknown how much of the improved result can be accredited to the adversarial training or is due to the effect of other architectural changes. Furthermore, this architecture is only implemented in a SISO environment, with the architecture's applicability in MA environments still unknown.

4 Experimental Setup

The same experimental setup is used for all experiments. Here we discuss the dataset, different system setups and the training protocol used.

4.1 Dataset

The training dataset is generated over a SISO channel using a simulated delay profile that is typical of traveling in a moving vehicle. Specifically, we use the Extended Vehicular A model delay profile [18] and 50Hz maximum Doppler shift, similar to Zhao et al. AWGN is added after the data has been transmitted over the channel to model receiver noise. We use the LTE toolbox in Matlab® to simulate the transmission used. The transmitted OFDM dataframe, containing 16 quadrature amplitude modulated quadrature and in-phase (QI) symbols, has a bandwidth of 10MHz and uses short cyclic prefixes. For the initial training set, no link-level techniques are used to ensure that the channel impairments are not obscured.

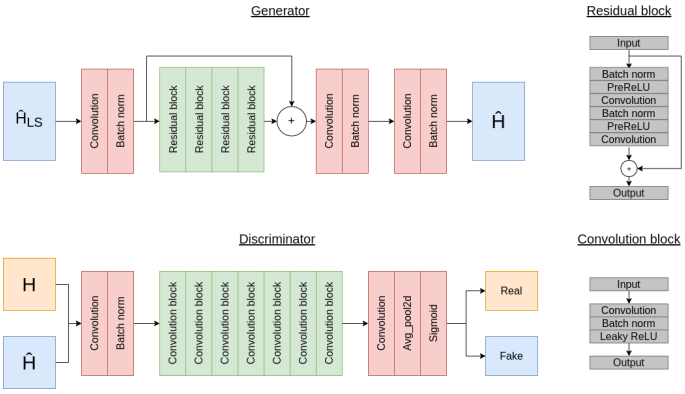


Fig. 2. Illustration of the modified SRGAN network structure proposed by Zhao et al. [5].

We provide the LS estimation $\hat{H}_{LS}$ of the CSI over a set sample size for the inputs to the network. The target of the modified SRGAN generator is the CSI before AWGN is added; an example of CSI being translated to a data sample can be found in Figure 3. We train and evaluate the networks using a 20dB AWGN range starting at 1dB and ending at 20dB SNR. The inputs and targets are represented using three features extracted from the demodulated OFDM signals: the absolute value of the QI symbols, the normalised quadrature component and the normalised in-phase component. We note that we have a three-channel representation in two dimensions, making our samples similar to RGB images presented to CNN architectures. We train the architectures using a training dataset of 40 000 samples and a validation set of 10 000 samples. An unseen test set for the SISO environment containing 10 000 samples is also generated and used for analysis in the input sample size selection and adversarial network training Sections. In this research, we apply several antenna diversity techniques to measure the performance of different CSI estimators. Due to each technique having a different number of channel paths that interfere with each other at the receiving antennas, the same dataset cannot be used for each method. However, each technique's evaluation uses identical simulation setups and consists of 400 data frames transmitted using the same channel and data generation seeds.

4.2 System Description

The neural networks used are derived from the SRGAN architecture discussed in Section 3. Thus, the generator is the same as the residual network (ResNet)

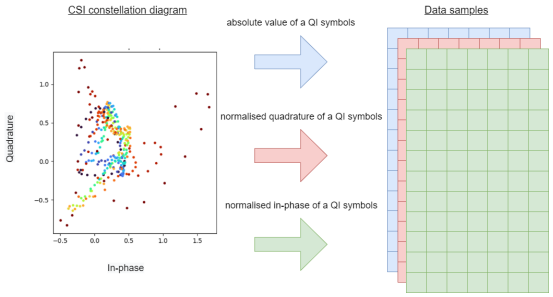


Fig. 3. An illustration of an CSI constellation diagram with 10dB antenna noise being translated to an input data sample with three features. Each QI symbol translates to a 1x3 point in the data sample and the data sample dimensions correspond to the sub-frame size selected, e.g. 12 x 7.

architecture used by Zhao et al., with four residual blocks, but reverts back to element wise MSE for the content loss. We select this architecture as it is competitive with other architectures in the field, as well as top performing statistical methods [5]. When we apply adversarial training, the discriminator consists of a VGG-like architecture with an output that can indicate if a real or fake sample has been presented. Both normal element-wise adversarial training and feature-based adversarial training are used to train networks, by either passing CSI or CSI features to the discriminator. The implemented feature-based adversarial training uses a pre-trained VGG-19 network with the decision layers removed as a feature extractor. We thus have a single architecture but three different training approaches: ResNet with no adversarial training, ResNet with a discriminator and ResNet with an additional feature extractor connected to its discriminator. We refer to these networks as ResNet, adversarial ResNet and SRGAN, respectively.

4.3 Training Protocol

All networks are trained using the Adam optimiser [19], with generators using MSE loss to ensure CSI estimation points are as close to the targets as possible. The discriminators use BCE loss, which, when used in conjunction with the sigmoid layer, gives the discriminator the ability to output the real or fake prediction. Finally, all networks are trained to convergence and final models for each run are selected using early stopping (selecting the model from the training run with best validation accuracy).

Due to the number of trainable hyperparameters and the computational expense of training adversarial networks, we only search over generator learning rate {1e-3, 1e-4, 1e-5}, discriminator learning rate {1e-3, 1e-4, 1e-5, 1e-6}, the

size of the discriminator's contribution to the gradient update {1e-2, 1e-3, 1e-4} and the amount of added noise to samples before being sent to the discriminator {noise variance: 0, 1} to use in the hyperparameter grid search. The best hyperparameter combination is selected by evaluating the loss of the trained network on the validation set. Based on the results of this hyperparameter sweep, we generate two additional networks of the best hyperparameter combination using different initialisation seeds. All results reported in Section 5 are an average of the results over these three seeds. This hyperparameter search protocol is performed for every network reported on in Section 5.

5 Analysis

As the first analysis step, we select an input sample size for the networks by comparing the BER for the different sample sizes. After choosing an input sample size, adversarial networks are trained and compared in a SISO environment. The same networks are then applied to different antenna diversity environments and analysed to determine if adversarial training has affected the BER performance of these systems.

5.1 Sample Size Selection Using ResNet

Input sample size is selected by creating a baseline using the ResNet architecture without adversarial training. To ensure that an adequate sample input size is used, we train three networks with different input sizes: 12x7, 24x14 and 120x14. We train these networks using the training protocol described in Section 4.3 to obtain the best-performing networks. Finally, the best performing networks' performance is compared to the LS implementation given as input to the NN and plotted against each other using percentage increase of correctly classified bits (PICCB) as metric. PICCB is calculated by dividing the rate of correctly classified bits from one training method with another.

From the results shown in Figure 4 we observe that the use of larger samples produces better performance than the use of smaller samples over the entire BER range, but that the increase in performance from 24x14 to 120x14 becomes small. We note that the ResNet generator provides the highest performance gain over LS at lower SNRs, where CSI is notoriously difficult to predict due to noise on the pilot signals masking true channel conditions.

From these results, we select the 24x14 sample size as input for the following sections, as the training and inference time is lower than the larger 120x14 sample size while not sacrificing significant estimation performance. This decision can further be motivated as the two best-performing input sample sizes display nearly identical behaviours over the SNR range.

5.2 Adversarial Network Training

Comparing the results of adversarially trained networks to non-adversarially trained networks in a SISO environment, we analyse the best performing net-

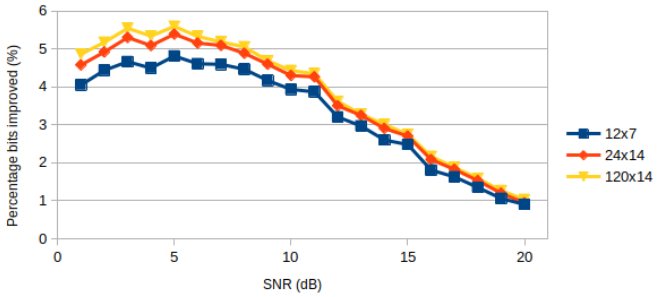


Fig. 4. PICCB of the CSI generated by three ResNet networks utilising different input sample sizes with respect to LS CSI estimation. This analysis is conducted over 20dB of AWGN using the unseen test set in a SISO environment.

works from each training method. We compare the results of the adversarial ResNet and SRGAN networks to the ResNet network. We observe the average loss over the entire test set for each network, followed by the average BER and, finally, the average PICCB. These metrics are displayed in Table 1 for each training approach and are averaged over three network seeds, as described in Section 4.3.

Table 1. Performance metrics of three generators utilising different training methods in a SISO environment. The PICCB metric is calculated with respect to ResNet (the non-adversarially trained network) over the entire SNR range.

	ResNet	Adversarial ResNet	SRGAN
Loss	**1.71e-2**	1.76e-2	1.73e-2
BER	1.42e-1	**1.41e-1**	1.41e-1
PICCB	0	**-1.20e-2**	-9.00e-3

The results presented in Table 1 do not show any significant difference between the training architectures, thus making adversarial training unnecessary. We plot the PICCB of the adversarial ResNet network and SRGAN with respect to ResNet over the SNR range in Figure 5, to further understand the effects of adversarial training. From this figure, we observe that the increase and decrease in performance are of small scale and seemingly random. From the results in Table 1 and Figure 5 we conclude that no advantage is obtained when equalising transmitted data using adversarially trained network CSI estimations in the considered SISO environment.

We note that these results contradict the results of Zhao et al. as they report on obtaining increased performance from using adversarial training. It is, however, difficult to measure the improvement proven by adversarial training in their work as the comparisons between networks are made using an upsampling method and the current implementation of SRGAN as opposed to non-adversarial and adversarial training methods.

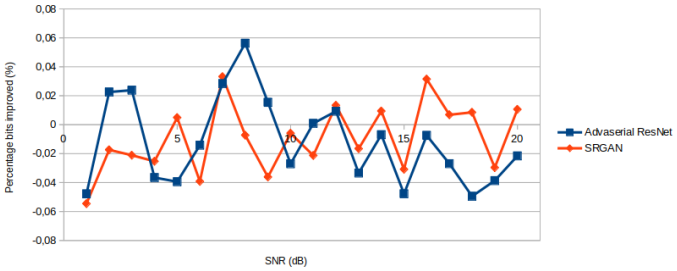


Fig. 5. PICCB of the CSI generated by adversarial training methods with respect to the trained ResNet CSI estimators. This analysis is conducted over 20dB of AWGN using the unseen test set in a SISO environment.

5.3 Receiver Diversity

We now move on to MA setups to evaluate the performance of the adversarially trained network in common wireless communication systems. The receiver diversity environment is analysed by implementing two systems, the first with two receiver antennas and the second with four receiver antennas. Both of these systems utilise the EGC algorithm to equalise the received datastreams. Using the same NNs trained in the previous sub-section, we evaluate the PICCB of the adversarial networks with respect to the ResNet network architecture for receiver diversity implementations. Examining the average PICCB over the SNR range in Table 2 we see that both adversarial training methods outperform the traditional trained ResNet, with the adversarial ResNet performing the best of the two networks. We also note that the BER performance increases as the number of receivers increases.

Table 2. PICCB over the entire test set SNR range of adversarially trained network with respect to non-adversarially trained ResNet, for a receiver diversity system.

Number of receiver antenna	Adversarial ResNet	SRGAN
2	**1.02e-1**	2.56e-2
4	**1.38e-1**	5.65e-2

Plotting the PICCB over the SNR range in Figure 6 we confirm this observation as the adversarial ResNet consistently outperforms the traditional ResNet over all trained SNRs in the test set. However, the same can not be claimed for the SRGAN implementation as it performs slightly worse than traditional ResNet at lower SNRs regardless of its average PICCB score. Based on these results it is hypothesised that adversarially trained networks produce CSI predictions that have different characteristics from predictions made by non-adversarially trained networks. These characteristics are not identifiable by the MSE loss function as the loss performance of all networks is similar in a SISO environment.

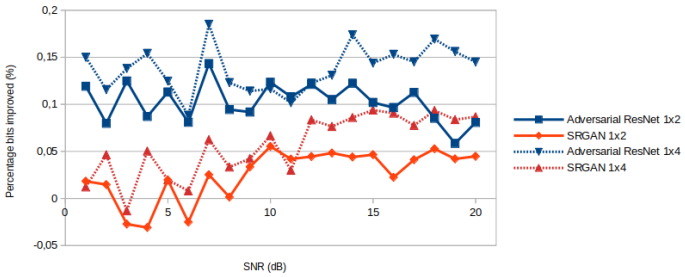


Fig. 6. PICCB of the CSI generated by adversarial training methods with respect to the trained ResNet CSI estimators. This analysis is conducted over 20dB of AWGN in a multiple receiver environment utilising the EGC method.

5.4 Transmit Diversity

Continuing the MA implementations analysis, we compare the adversarially trained models to the non-adversarially trained models in a transmitter diversity environment. Using two transmit antennas, we construct two environments, one with a single receiving antenna and the second with two receiving antennas. For transmitter diversity, we implement Alamouti OSTBC and once again compare the adversarial networks to the traditionally trained ResNet. Examining the PICCB of this comparison in Table 3 we once again obtain results that differ from the results in the SISO environment. Firstly we observe that the adversarial ResNet no longer outperforms the traditional ResNet. Secondly, we observe that the SRGAN outperforms adversarial ResNet, but not traditional ResNet, again hinting at the possibility that these identical architectures predict differently because of their training method.

Table 3. PICCB of the entire test set SNR range of adversarially trained network with respect to non-adversarially trained ResNet, for a transmitter diversity system.

Number of receiver antenna	Adversarial ResNet	SRGAN
1	**-1.93e-2**	-1.85e-3
2	-3.66e-2	**-1.06e-2**

Further examination of the PICCB over the SNR range of the test set, shown in Figure 7, displays how SRGAN outperforms adversarial ResNet at higher SNR levels. SRGAN, however, does not outperform traditional ResNet for most of the SNR range. While adversarial ResNet does compete with SRGAN at lower SNRs, this advantage is lost when the PICCB declines as the SNR increases.

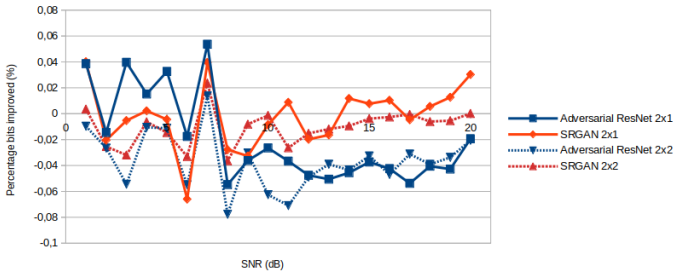


Fig. 7. PICCB of the CSI generated by adversarial training methods with respect to the trained ResNet CSI estimators. This analysis is conducted over 20dB of AWGN in multiple receivers and transmitters environments utilising the Alamouti method.

Once again, we observe a difference in performance from networks that seemingly performed the same in the training phase of this analysis. These differences are of such a small scale that they are comparable to the differences in the SISO channel. However, in this instance, traditional ResNet consistently performs the worst of the three networks and SRGAN still performs similarly to traditional ResNet at higher SNRs.

From this analysis of different MA environments, we see that adversarial training provides unexpected increases and decreases in performance. Adversarial training seems to introduce characteristics into predictions that affect different MA environmetns in different ways. These results further lead us to believe that the performance indicators used in the training phase are inadequate in selecting networks for MA CSI estimation tasks.

6 Conclusion

This paper implements three methods of training using the same generator network to estimate high-quality CSI from low-quality LS estimations in an LTE environment. We ensure that an adequate input sample size is selected for our baseline non-adversarially trained network by considering three input sample sizes. When analysing the results of our ResNet implementation as baseline we observe:

- improvement over the low-quality LS input over the entire SNR range.
- the ResNet architecture achieves the most BER gain at lower SNR levels.
- the 24x14 input sample performance closely matches the performance of the larger 120x14 sample size.

All training methods perform similarly in the SISO training environment. This behaviour would typically be seen as an indication that the networks will perform similarly in MA environments, as the best-performing networks are selected based on MSE performance in SISO networks. When introducing MA environments, however, we find that:

- adversarially trained networks can consistently outperform traditional non-adversarial methods in receiver diversity cases.
- in the transmitter diversity cases, the adversarial ResNet shows a decrease in performance.
- in the transmitter diversity cases, the SRGAN using feature maps contended with traditional ResNet at higher SNRs.

The difference in network performance between the SISO training environment and MA deployment environment indicates that when the environment of deployment and the effects of adversarial training is well understood, increased BER performance can be obtained. We obtain this increase in performance at the same number of floating point operations as non-adversarially trained networks.

Acknowledgements The authors gratefully acknowledge the financial support of this study by the Telkom CoE at the NWU and Hensoldt South Africa. The authors acknowledge the Centre for High Performance Computing (CHPC), South Africa, for providing computational resources to this research project.

References

1. "Evolved universal terrestrial radio access E-UTRA; base station BS conformance testing," 3GPP TS 36.141, July 2018.
2. P. Sure and C. M. Bhuma, "A survey on OFDM channel estimation techniques based on denoising strategies," *Engineering Science and Technology, an International Journal*, vol. 20, no. 2, pp. 629–636, 2017.

3. E. Karami, "Tracking performance of least squares MIMO channel estimation algorithm," *IEEE Transactions on Communications*, vol. 55, no. 11, pp. 2201–2209, 2007.
4. O. Edfors, M. Sandell, J.-J. Van de Beek, S. K. Wilson, and P. O. Borjesson, "OFDM channel estimation by singular value decomposition," *IEEE Transactions on Communications*, vol. 46, no. 7, pp. 931–939, 1998.
5. S. Zhao, Y. Fang, and L. Qiu, "Deep learning-based channel estimation with SR-GAN in OFDM systems," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1–6.
6. Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *MIMO-OFDM wireless communications with MATLAB*. IEEE Press, 2011.
7. D. F. Carrera, C. Vargas-Rosales, N. M. Yungaicela-Naula, and L. Azpilicueta, "Comparative study of artificial neural network based channel equalization methods for mmWave communications," *IEEE Access*, vol. 9, pp. 41 678–41 687, 2021.
8. N. Soltani, H. Cheng, M. Belgiovine, Y. Li, H. Li, B. Azari, S. D'Oro, T. Imbiriba, T. Melodia, P. Closas *et al.*, "Neural network-based OFDM receiver for resource constrained IoT devices," *arXiv preprint arXiv:2205.06159*, 2022.
9. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.
10. C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the ICVPR*, 2017, pp. 4681–4690.
11. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
12. K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the ICVPR*, 2015, pp. 1026–1034.
13. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
14. C. Luo, J. Ji, Q. Wang, X. Chen, and P. Li, "Channel state information prediction for 5G wireless communications: A deep learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 227–236, 2018.
15. Y. Yang, F. Gao, X. Ma, and S. Zhang, "Deep learning-based channel estimation for doubly selective fading channels," *IEEE Access*, vol. 7, pp. 36 579–36 589, 2019.
16. E. Balevi and J. G. Andrews, "Deep learning-based channel estimation for high-dimensional signals," *arXiv preprint arXiv:1904.09346*, 2019.
17. Y. Yang, Y. Li, W. Zhang, F. Qin, P. Zhu, and C.-X. Wang, "Generative-adversarial-network-based wireless channel modeling: Challenges and opportunities," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 22–27, 2019.
18. "Evolved universal terrestrial radio access E-UTRA; user equipment (UE) radio transmission and reception," 3GPP TS 36.101, July 2018.
19. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.