Beam selection using Machine Learning in Massive MIMO systems

Simon Persson
si6426pe-s@student.lu.se

Supervisors: Harish Venkatraman Bhat (Ericsson)
William Tärneberg (LTH)

Examiner: Maria Kihl

September 2023

Abstract

In mobile communications the demand for increased capacity and speeds is a constant. 5G NR is the latest generation which is now seeing widespread adoption. Crucial technologies like Massive MIMO and beamforming are enabled by the use of large antenna arrays. While these arrays allow a 5G network to handle larger amounts of data at higher speeds they do have drawbacks. Specifically, the power requirements associated with using many large arrays are sizable.

Antenna selection is a potential solution to this problem, allowing the antenna array to use only a subset of elements at off-peak times. Selecting the most favorable beams for transmission and reception is an important step towards achieving this goal. The aim of this thesis is therefore to apply machine learning techniques to this problem in order to predict the locations of optimal beams.

Throughout the testing process several machine learning models and approaches to solving the problem were explored. In each step of the process the prediction accuracy was evaluated and improvements were made. Several of the models displayed the ability to make accurate predictions, which will aid in solving the antenna selection problem.

The highest accuracy was achieved when predicting one of two predefined beam regions based on Precoding Matrix Indicator-values using a Deep Neural Network model. This test yielded a prediction accuracy of 87.4%.

Acknowledgments

I would like to give thanks to Ericsson for giving me the opportunity to conduct this master's thesis and learn many great things about an interesting topic. I am grateful to all all the wonderful people who have helped me along the way. Special thanks go to Harish Venkatraman Bhat for guiding me through the process and helping me understand important concepts.

I also express my gratitude to the friends and family who have supported me not just during my thesis, but throughout my time at LTH.

Popular Science Summary

The fifth generation of mobile communications brings great improvements to speed and capacity compared to its predecessors. While this may be a significant advantage of 5G there are drawbacks like increased power consumption. Antenna selection looks to be a promising solution to this problem.

A crucial part of any 5G system is the use of large antenna arrays consisting of a large number of smaller antenna elements. These elements work together and allow the capacity across a radio link to be multiplied.

Beamforming is an important technique enabled by the use of antenna arrays. The phase and amplitude of a signal can be modified at each antenna element in such a way that constructive interference is created and aimed in a specific direction, also known as a beam. Using all antenna elements for this process allows transmissions to be focused while reducing interference for received signals. The downside of this is high power consumption. For this reason this project works towards finding a way of using only a subset of the available antennas. An important step towards achieving this goal is beam selection, or selecting the best beams for a given signal.

This project has investigated ways of selecting the most suitable beams using a variety of machine learning techniques. In a set of received beams the suitability of each one can be determined based on amplitude, meaning the higher the power the better the beam. The aim of this project is therefore to predict which beams will have the highest power based on a variety of input variables.

Several approaches to the problem and machine learning models were tested over the course of the thesis. The most promising results were achieved when dividing the set of beams into predefined regions and predicting which has the highest power beams. Making these predictions based on so PMI data, which is used by the base station to enable multiple data streams, yielded the highest accuracy.

In the future this work will have to be expanded upon to enable antenna selection, however having a reliable way of selecting beams is a crucial step

towards achieving this goal.

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface.

CQI Channel Quality Indicator.

CSI Channel State Information.

DNN Deep Neural Network.

KNN K-nearest neighbors.

LTE Long Term Evolution.

MAE Mean Absolute Error.

MIMO Multiple Input, Multiple Output.

ML Machine Learning.

NR New Radio.

OFDM Orthogonal Frequency Division Multiplexing.

PMI Precoding Matrix Indicator.

ReLU Rectified Linear Unit.

RI Rank Indicator.

SRS Sounding Reference Signal.

UE User Equipment.

CHAPTER 1

Introduction

1.1 Background and Motivation

Mobile communication technologies are an essential part of everyday life. Thanks to the advancements made over the course of several decades staying connected is more important, and easier, than ever. In recent years, LTE (Long-Term Evolution) has become the standard for wireless communication used around the world. However, as the needs for greater bandwidth and faster communications continue to grow the limitations of LTE have become increasingly evident. The frequencies used in LTE are often insufficient to meet the high bandwidth requirements of today.

5G NR is meant to address these issues. A higher bandwidth is achieved by operating in a higher frequency range and through the use of a broader spectrum larger amounts of data can be handled. Another important technology enabling the use of NR are large antenna arrays consisting of several connected antennas working together through massive MIMO, allowing for increased link capacity. Beamforming is another crucial technology enabled by the antenna array which allows transmissions to be directed while reducing the interference of received signals.

While the large antenna arrays used in 5G NR provide many benefits, they also consume a lot of power. Using all antennas to transmit and receive data at all times is wasteful, especially when traffic load is low. Antenna selection has the potential to be an effective solution to this problem. By using only a subset of antennas at off-peak times power consumption could likely be reduced. An important step towards achieving this goal is the selection of beams received by the antenna array. That is the focus of this thesis.

1.2 Thesis objectives

The original purpose of this thesis was to develop an antenna selection algorithm utilizing machine learning. Testing this algorithm in a simulation or real world environment was another aim. However, the focus shifted when investigating the problem of beam selection. Throughout the thesis several different approaches to this problem had to be tested while their viability had to be evaluated. For this reason solving the beam selection problem through the use of machine learning became the main focus of the thesis.

1.2.1 Main questions

- Q1: Is it possible to accurately predict a subset of beams to use using a machine learning model?
- Q2: How should predictions be made and based on what input data?
- Q3: Which machine learning model yields the highest prediction accuracy?

During the testing process different approaches to making predictions were tested and several machine learning models were evaluated. This was done through an iterative process were issues with the current methodology were assessed. Based on these assessments adjustments to the machine learning model or testing approach were made.

1.2.2 Scope and limitations

As mentioned above the original aim of the thesis was to develop an antenna selection algorithm and testing it, however this ended up being outside the scope as a greater focus was placed on beam selection. This work done in this thesis is limited to beam selection only.

There are a great number of machine learning techniques that could have been investigated as possible means of performing beam selection. During the thesis however, adjustments often had to be made to experiment methodology in order to produce useful results. For this reason less focus was placed on testing many different machine learning models. In this thesis testing is mainly limited to two types of models, linear regression and deep neural networks. A K-nearest neighbors approach was also tested briefly.

1.2.3 Tools used

All machine learning models, log parsing scripts and other scripts were implemented using Python. Plots of data were created using the matplotlib library. The frameworks used to implement the various ML-models used during this thesis are detailed in the 'Machine Learning models' section.

1.3 Report structure

Initially, the literature study which was conducted is described and the results of various related work is presented and discussed.

This report then provides an introduction to 5G NR and dives deeper into the technology concepts required to understand this thesis. This is followed by explanations of commonly used technical terms and the theory behind them.

The data used by the machine learning models tested is described in the following section. Calculations made using this data are also presented.

This is followed by descriptions of the machine learning models used throughout this thesis and the theory behind them. The frameworks used to implement these models are also described.

As mentioned several iterations of testing methodology and ML-models are present in this thesis. The details of each iteration and the models used during each are described in the next section.

Results and the discussion of those results follow. Here the issues with each iteration are also described in detail.

Finally conclusions and potential future work are presented.

CHAPTER 2

Related work

Searching for and investigating other work closely related to this thesis was an important process that helped focus the thesis work while also placing its results in a broader context. Initially, the purpose of the literature study was to get a better understanding of the massive MIMO and beamforming technologies used in the 5G NR system. This was done by looking at a variety of sources, including the research papers listed in this section. Later, papers related to different antenna selection methods and their effectiveness were also investigated. Finally, work focusing on using machine learning techniques for antenna selection was reviewed in order see what approaches to the problem had previously been tried and how effective they were.

2.1 Massive MIMO and Beamforming

Massive MIMO and Beamforming are well researched technologies which have been proven effective and are used in 5G NR. Relevant technical details are presented in the '5G NR Overview' chapter.

In "Massive MIMO for Next Generation Wireless Systems" Larsson et al. [9] present an overview of key concepts in the massive MIMO technology. They also evaluate contemporary research on the topic in order to draw conclusions regarding the potential of the technology. It is concluded that massive MIMO offers large improvements in terms of efficiency and reliability, making it optimal for use in technologies beyond LTE, such as 5G.

Ali et al. [7] also provide an overview of massive MIMO in "Beamforming techniques for massive MIMO systems in 5G: overview, classification, and trends for future research". The main focus of the paper however, lies in beamforming and investigating its benefits. A summary of the technology is provided while

different beamforming techniques are evaluated. Finally, the authors present an optimal method of beamforming which is achieved by combining several different techniques.

The details of different beamforming techniques lie outside the scope of this thesis, however the explanations of the key concepts behind the technology and its benefits are highly relevant.

2.2 Antenna selection

The concept of using only a subset of antennas in a Massive MIMO system is one that has been thoroughly investigated. Several different methods of achieving this goal have been tested and evaluated.

Gao et al. [10] present and evaluate an antenna selection method based on a branch-and-bound search algorithm where antenna elements are represented in a search tree. This algorithm is then compared to using the entire antenna array in a simulation environment. The testing shows that the antenna selection algorithm is able to achieve similar performance compared to the baseline tests. It is concluded that antenna selection is a promising and practical technology for use in massive MIMO systems.

Another antenna selection algorithm is presented by Siljak et al. [11]. The authors present a method for use in distributed massive MIMO systems. An algorithm based on petri nets in which antenna elements are divided into groups is evaluated. These elements are then allowed to move between between these groups. The proposed algorithm is compared to other selection methods and shows comparable performance, however the novel solution excels in scenarios where a large number of users are present.

Wu et al. [19] compare two different antenna selection algorithms and evaluate their respective performance. An optimal but complex algorithm is compared to one with reduced computational complexity in a simulation environment. These simulations show that a channel capacity close to that of the optimal algorithm can be achieved with the simpler algorithm.

While antenna selection algorithms based on techniques other than machine learning are not evaluated in this thesis the results of the studies above are still interesting. It is clear that antenna selection is a viable concept and that a variety of approaches to the problem are possible.

2.3 Antenna selection using machine learning

Performing antenna selection using machine learning techniques is an alternative solution which has also been investigated previously. Yu et al. [22] propose a solution based on multi-label learning where a deep neural network is implemented to predict a subset of antenna elements. Tests are conducted in a simulation environment where the ML-based algorithm is compared to more traditional antenna selection scheme. The results show that using a machine learning model

yielded comparable capacity across the radio link while decreasing computation time.

Deep neural networks are also investigated by Meenu Khurana [14]. An antenna selection scheme for vehicle-based MIMO systems is proposed and evaluated. This scheme is implemented using a convolutional deep neural network and compared to a more conventional algorithm. In simulations this DNN-model was also shown to achieve higher capacity while decreasing delays.

A different machine learning approach which is also based on neural networks is presented by Vu et al. [21]. In this paper the algorithms examined handle both antenna selection and power allocation, leading to higher computational complexity for traditional algorithms. By employing a machine learning based approach this complexity can be vastly reduced. This was verified during testing as the ML algorithm reduced execution time while maintaining 90% of the traditional algorithm's performance.

While power allocation lies outside the scope of this thesis it is interesting to note that the machine learning approach has been shown to produce similar performance compared to other algorithms.

Another study supporting the viability of a deep neural network approach was conducted by Zhong et al. [23]. Similarly to previously mentioned studies, a DNN model is implemented and used to predict antenna elements. The performance of this model is then compared to that of a conventional antenna selection algorithm as well as no antenna selection. While similar performance was achieved with the two antenna selection algorithms using the DNN model yielded a large increase in channel capacity when compared to using no antenna selection.

The approaches to the antenna selection problem and the design of the machine learning models tested during this thesis differ from those used in the work discussed above. However, prior evidence of antenna selection using machine learning being viable is important context. It is also interesting to note that using Deep Neural Networks yielded good results in several studies. This is part of the reason why the DNN was one of the models investigated during this thesis.

CHAPTER 3

5G NR Overview

This chapter provides a short overview of the 5G new radio concept and dives deeper into some of the aspects most relevant to this thesis.

3.1 Introduction to 5G NR

The fifth generation of mobile communication technology has been in development for several years, however more recently the technology is seeing widespread adoption. As with its predecessors a 5G network is a cellular one, meaning geographical areas are divided into "cells" served by at least one base station. Compared to LTE, 5G features a much broader frequency spectrum, allowing for an increase in speeds while increasing the amount of data that can potentially be handled by the network [6]. The efficient use of spectral resources in a 5G NR network is achieved through the use of multiple antenna technologies, namely multi-user massive MIMO.

3.2 Massive MIMO

Massive MIMO is a technology that is crucial to the 5G NR system. Regular MIMO (multiple input, multiple output), a method that is commonly used in radio, works by using multiple transmission and receiving antennas. This way multipath propagation can be exploited, allowing the capacity of the radio link to be multiplied through spatial multiplexing. Massive MIMO is an extension of this concept that increases the number of antennas drastically. This is what the "massive" part of the name refers to. By expanding the antenna array spectral efficiency is increased, meaning more users can be serviced at any given time

[18]. Coverage is also improved as a stronger and more reliable signal can be provided.

3.2.1 The antenna array

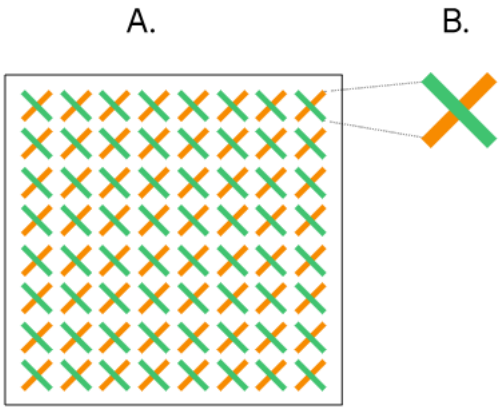

Figure 3.1: Illustration of an antenna array containing 128 cross polarized antenna elements. [8]

The large antenna array is an important part of massive MIMO. A total of 64 antennas are present in the array used to collect data during this thesis. The illustration in figure 3.1 depicts an array of 128 elements, however the concept remains the same. The antenna elements in the array used for testing are arranged in a grid consisting of 8 columns and 4 rows. In each grid slot two cross-polarized antennas are located, giving a total of 64 elements. Antenna cross-polarization helps reduce interference for received signals.

3.3 Beamforming

Beamforming is a crucial technique used in 5G NR for the purpose of focusing transmitted signals to a specific location while improving the signal-to-noise ratio of received signals. This is done by combining the elements of the antenna array in such a way that constructive interference is experienced by signals at a certain angle while others experience destructive interference. For example, when transmitting a signal, the directionality can be altered by controlling the phase and amplitude of the signal at each antenna element to create a desired interference pattern. Similarly, when receiving, data from all elements can be combined such that the desired pattern is observed.

3.3.1 Beam space vs antenna space

Data handled during this thesis consists of beams received by the base station antenna array. This means that each beam consists of data received from all antennas which has been combined to achieve the desired directionality. In other words, beam space data is created through the combining of antenna space data. As there are 64 antenna elements in the array beams from a total of 64 directions will be received at any given time. Some of these will see constructive interference while others will be destructive. The purpose of this thesis is to exploit this fact and attempt to predict where within the range of beams the highest amplitude can be found.

CHAPTER 4

Relevent technical terms

4.1 SRS Channel Estimates

An integral aspect of communication between a base station and a UE are sounding reference signals (SRS). Such a reference signal is transmitted by the UE in the uplink direction and contains the information needed for the base station to estimate the quality of the channel. The channel state information (CSI) which is estimated is comprised of a channel quality indicator (CQI), rank indicator (RI) and precoding matrix indicator (PMI).

The SRS is an orthogonal frequency division multiplexing (OFDM)-signal, meaning that one stream of information is sent across several subcarriers placed within close proximity of each other.

The SRS channel estimation data used in this thesis is downsampled and collected in groups of 64 values, representing each of the 64 directions. Details of the SRS signals handled in this thesis are presented in the "Data" section.

4.2 PMI

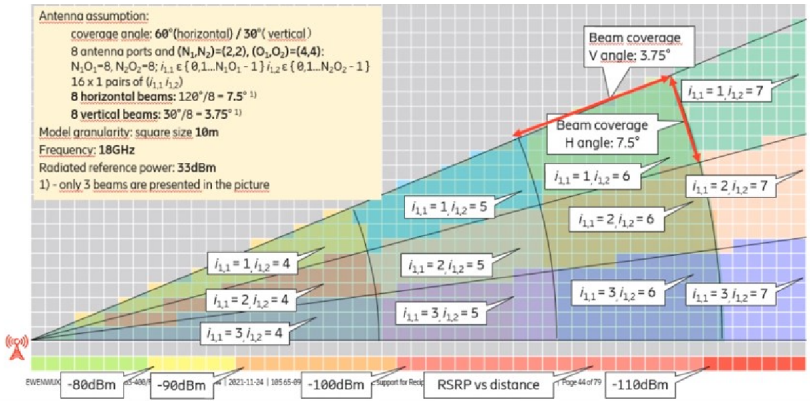

Figure 4.1: Illustration showing how PMI values correspond to distance from and angle to the base station. Source: Ericsson

PMI, or precoding matrix indicator is essentially a recommendation from a UE of which precoding matrix should be used for transmission. Precoding is a process by which multi-layer transmission is enabled. This is done through the application of one of several predefined matrices. PMI is therefore comprised of index values indicating which precoding matrix should be selected. Specifically, the PMI data handled during this thesis contains two indices. As shown in figure 4.1 these values also correspond to physical location, with the first index indicating the angle and the second indicating distance from the base station. These properties are exploited during this thesis for the purpose of making predictions for beam selection.

CHAPTER 5

Data

In this chapter the data collected to train the machine learning models is described. The calculation of beam power, which is the main metric used for beam selection, is also described. An overview of the system in which the data is used is also presented.

The process of collecting the input data used throughout this thesis was handled by employees at Ericsson. This work therefore lies outside the scope of this thesis, however a brief description of the collection methodology is provided below.

The work done in this thesis instead focuses on investigating which data sets to use, how to transform that data and how to integrate it into a beam selection system.

5.1 The datasets collected

Over the course of this thesis several datasets were collected. These are stored in log files which are interpreted by a log parsing script. During data collection signals sent from a UE to a base station were recorded. Various movement patterns were used during this process to bring some variation to the collected data. A total of four logs of various sizes were used throughout the testing process. The initial log file used, AndreAroundGarage.log, contains only SRS-data and is used as input for the models used in the initial experiments conducted. Subsequent logs contain PMI data interspersed among the SRS data previously used. These files are named SRS_and_PMI.log, PMI_SRS_around_watertower.log and PMI_SRS_around_garage.log and are used during later tests.

5.2 The log file

Each row of data within the log file is associated with a timestamp which can be used as an identifier. Lines of SRS data also contain a value to identify the specific grouping of down sampled channel estimations. The SRS data in the file is spread across four transmission layers, however to simplify data interpretation only data from the first layer (layer 0) was included in testing.

The SRS data in the log files contains information on how to modulate the I and Q phases of the sine wave, known as I/Q data. Within any group of channel estimates are 64 beams, each of which is represented by a 32 bit word where the first 16 bits represent the I phase and the last 16 represent the Q phase. By calculating the two's complement of these binary values the actual value of the phases can be retrieved. The log parsing script handles the interpretation of I/Q data and uses it to calculate beam power.

```
layer = 0, direction = 0-7: {a a a a a a a a}
layer = 0, direction = 8-15: {a a a a a a a a}
layer = 0, direction = 16-23: {a a a a a a a a}
layer = 0, direction = 24-32: {a a a a a a a a}
layer = 0, direction = 33-39: {a a a a a a a a}
layer = 0, direction = 40-47: {a a a a a a a a}
layer = 0, direction = 48-55: {a a a a a a a a}
layer = 0, direction = 56-64: {a a a a a a a a}
```

Figure 5.1: Visualization indicating how SRS data is formatted in the log files used during this thesis.

The structure of the relevant SRS data in the log file can be seen in figure 5.1, showing an example group of 64 I/Q values. In this visual representation every 'a' represents a hexadecimal datapoint with the format $0xAAAAAAAA$. These are converted to binary format in order to make the calculations described above easier. The 64 values in the SRS data group are spread across 4 lines in the log file. In addition to this the transmission layer is also indicated, allowing the log parser to filter out unwanted data.

5.3 Beam power

In the machine learning models used during this thesis beam power, or amplitude was the main metric used to determine which region should be selected for a given input. As the name suggests the beam power is a measurement indicating strength of a beam. The objective of the thesis is to select a subset of the antenna array given an input of 64 beams. Therefore, selecting a region based on where the strongest beams are located within the beam range was seen as a reasonable approach. The idea is that at a late stage the bit error rate could be minimized by including only strong beams. For a given beam within an SRS

channel estimate group the beam power can be calculated using the following formula:

$$P = H(b) * H'(b)$$

where P is the beam power, $H(b)$ is the complex I/Q value retrieved from the log file and $H'(b)$ is its complex conjugate. Beam power calculation is also handled by the log parsing script.

5.4 Beam regions

To eventually achieve the goal of only activating a subset of antennas at any given time a decision has to be made of which beams within the 64 beam range to use. In other words, a subset of beams has to be selected. The beam range is therefore divided into regions during several tests. The beam power values calculated are the metric by which a region is selected. During the classification problems investigated in this thesis each region is assigned an index which is used by the log parser to label input data.

5.5 PMI data

Three of the log files used include PMI data. Similarly to the SRS data each row of PMI values is associated with a time stamp, separate from any other data. In this row the two PMI index values are stored. The log parser reads these values and maps the following SRS rows to this PMI pair until new PMI values are encountered. This process is then repeated.

```
wbPmiI1[a, b, c]
```

Figure 5.2: Visualization indicating how PMI data is formatted in the log files used during this thesis.

The formatting of the relevant PMI data in the log file is visualized in figure 5.2. An array containing three values is used, but only the first two of these are used in this thesis as they represent the PMI index values.

5.6 System overview

The various types of data collected during this thesis are used in a system where a machine learning model is trained and used to make beam region predictions. A brief overview of this system is provided in this section. More detailed descriptions of testing steps, machine learning models and results are provided in subsequent chapters.

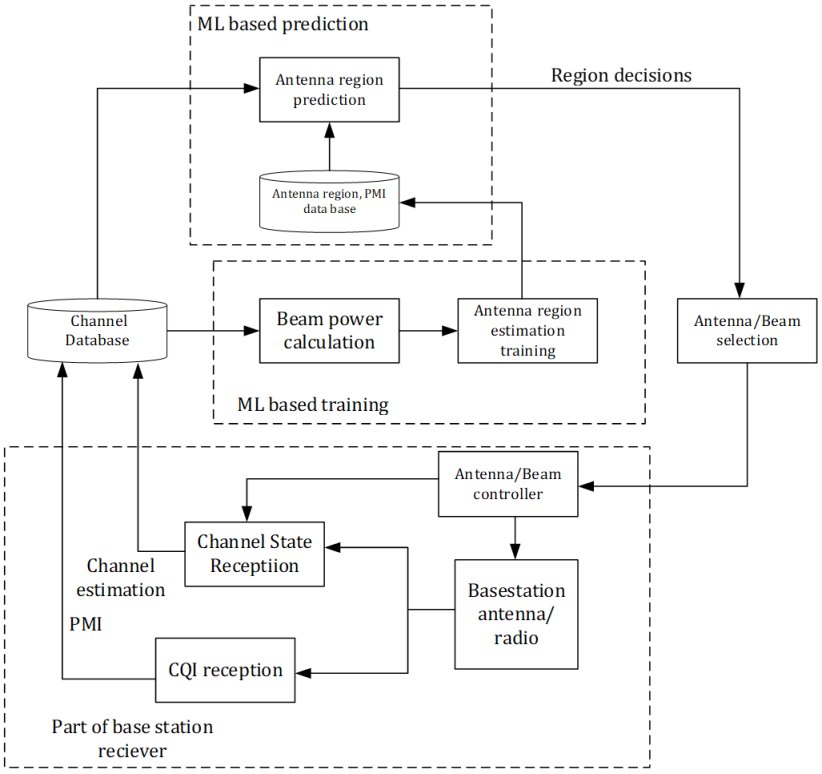

Figure 5.3: Overview of the system in which PMI data is used to make beam region predictions. Source: Ericsson

SRS Channel Estimate data and PMI index values are received from the base station receiver as show in figure 5.3. The machine learning model is trained using this data according to the methods described in the 'Experiment iterations' chapter. After training predictions can be made based on new input data to produce a beam region decision. The selected region is then fed back to the base station for use in its controller. The details of channel state reception, CQI reception, the antenna/beam controller and the base station antenna/radio lie outside the scope of this thesis. Their functions are therefore not described in this report.

CHAPTER 6

Machine Learning models

Evaluating different machine learning models and configuring these with appropriate parameters is one of the main areas of work in this thesis. Deciding which models to evaluate was included in this process.

Explanations of crucial machine learning terms used throughout this thesis are included in this chapter. In addition to this, the machine learning models implemented and used during the testing process are described as well as the reasons they were selected. The frameworks and their features which were used to implement them are also explained.

6.1 Machine learning terms

6.1.1 Loss

Loss is essentially a measurement of the quality of a models prediction. A perfect prediction would have a loss of 0 and the higher the loss of a single prediction the worse it is. The aim when training a machine learning model is to minimize this value. Loss can be calculated in many different ways according to one of several loss functions. What function is optimal depends on which model is used [20]. Throughout this thesis several different loss functions were tested for each model, and whichever gave the best result was selected. The different functions used are described below.

Mean absolute error

The mean absolute loss function is a very simple loss function to compute, but it can be applied in a lot of cases. It calculates the difference between the

output value predicted by the model and the actual label value according to the following formula [20]:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|Y_{predi} - Y_{labeli}|$$

where N is the number of values predicted, Y_{pred} is the prediction and Y_{label} is the label value. MAE is the mean absolute error.

Categorical and binary crossentropy

Another loss function used in some tests is the categorical crossentropy loss function. Crossentropy essentially measures the the difference between two probability distributions [3] and is calculated according to the following formula:

$$H(P, Q) = -\sum_{x \in X} p(x) \log q(x)$$

where P and Q are two probabilty distributions. In a prediction task, the target distribution P for a certain input is known as the label is known. In other words, if an input is present in the training dataset its label is certain. The goal of the model is to approximate the probability distribution Q which indicated the probability of each class label for a given input [3]. The goal is to minimize the crossentropy when training the model. Categorical crossentropy is a loss function for a multi-class classification task while binary crossentropy is used for binary classification. Crossentropy loss functions are commonly used for multilabel classification tasks [16].

6.1.2 Activation functions

An activation function is a function which is applied to weighted inputs in a neural network node. The purpose of the activation function is to aid the network in understanding complex pattern in the input data [13]. The most ability to add non-linearity to a network is a crucial feature of the function.

There are many different activation functions which can be used in a network. Below are explanations of the ones used in this thesis.

Linear activation

A linear activation can more accurately be referred to as "no activation" [1]. A node using linear activation does not do anything to the weighted input it receives, it simply passes it to the next node. Linear activation is utilized in the linear regression models used during testing.

Rectified linear unit activation

$$f(x) = max(0, x)$$

The rectified linear unit activation, or ReLU function introduces an element of non-linearity to a neural network[1]. It does so by outputting the input received by the node if that input is positive, however it outputs 0.0 if it is negative. This means that neurons in a layer with a ReLU activation function will only be activated if their input is positive. Therefore the ReLU activation function is more computationally efficient compared to some other activation functions where all nodes are activated at all times. For these reasons the rectified linear unit activation function has become the default choice for many neural networks, and it is commonly used in the models tested during this thesis.

Sigmoid activation

The sigmoid activation function is one of the most commonly encountered in neural networks. A defining feature of the sigmoid function is that it provides a smooth gradient, meaning sudden jumps in output values are less likely to occur [1]. For this reason it is commonly used in the ouput layer of a neural network. The formula for the sigmoid activation function is as follows:

$$f(x) = \frac{1}{1 + e^{-x}}$$

where x is the neuron's input. The larger the value of x the closer the output will be to 1.0. The opposite is true for negative inputs where the output will converge to 0.0.

6.2 Machine learning models used

Two main machine learning models were used throughout the tests conducted during this thesis. Linear regression and Deep neural networks are the main model types whose viabilities were investigated. In the final experiment a K-nearest neighbors method was also evaluated.

The linear regression model was selected as an initial starting point on which to build. Linear regression is a simple model which is easy to implement and could therefore be used to quickly obtain inital results.

Deep Neural Networks are more complex models which can be applied to a variety of scenarios, such as the classification problems enctountered in this thesis. As a DNN provides more opportunities for configuration and customization, creating potential for higher prediction accuracies. For this reason the DNN was selected for evaluation alongside linear regression.

Finally, a KNN-classifier was briefly tested for comparison with the other models. KNN is a simple and commonly used model that can be applied to classification problems, which is why it was selected.

6.2.1 Linear regression model

Linear regression is a supervised learning method, meaning it requires labeled datasets to train on. It works by attempting to fit a linear regression line to a set of training data points in such a manner that the average loss is minimized. The prediction of the trained linear regression model is calculated according to the following formula:

$$y = c + \sum_{i=1}^{N} m_i x_i$$

where y is the prediction, c is a constant m_i is a coefficient determined during the training process and x_i is input feature i. N is the number of input features.

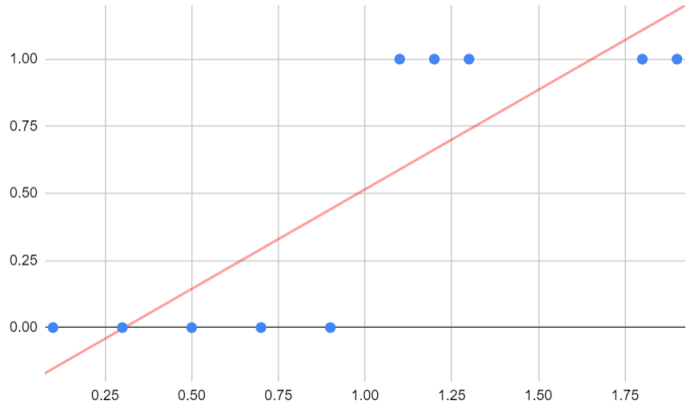

Figure 6.1: Visualization of an example linear regression model with a single input feature used for binary classification.

A simple linear regression model with a single input feature is visualized in figure 6.1. Here, a regression line has been fitted to a set of data points labeled either '0' or '1'. When a linear regression model is used for classification tasks the predicted class can be obtained by rounding the model's prediction to the nearest integer. This is the method used during this thesis.

6.2.2 Deep Neural Network

All neural networks are constructed using several layers. Each of these layers is made up of one or several nodes, also referred to as neurons or perceptrons. Each node has one or more weighted inputs, an activation function which is applied, and one or several outputs [2]. Through the training process the optimal weights throughout the network are determined.

Every network has an input layer consisting of the same number of nodes as the number of input features. The output layer similarly consists of several nodes corresponding to the number of outputs. Between these a regular neural

network has one hidden layer containing several nodes. In each node a predetermined activation function is applied and weights are adjusted during the training process.

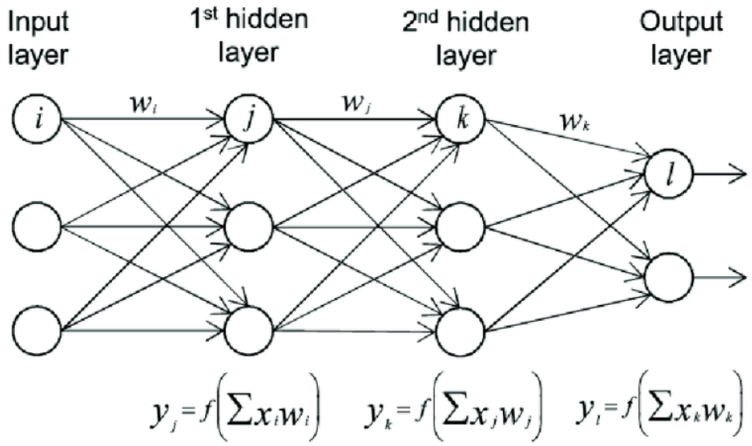

Figure 6.2: Illustration of a deep neural network with two layers.[17]

A deep neural network includes two or more hidden layers, as illustrated in figure 6.2. In each layer the output y is determined by multiplying weights w_i with inputs x_i and applying an activation function f. As most of the classification problems in this thesis have a relatively small number of inputs and outputs only two hidden layers were used in all DNN models tested. A rule-of-thumb for creating deep neural networks is that the number of nodes in a hidden layer should be somewhere between the number of input nodes and the number of output nodes [15]. This rule was followed when designing the networks used in this thesis.

6.2.3 K-nearest neighbors classifier

The K-nearest neighbors, or KNN algorithm was tested in one of the later test iterations. KNN is a simple but commonly used supervised algorithm. It classifies an input by comparing it to a known labeled dataset. The algorithm calculates the distance between the input and the surrounding data points. Classification is done by evaluating the k nearest neighboring points and selecting the most common label [12].

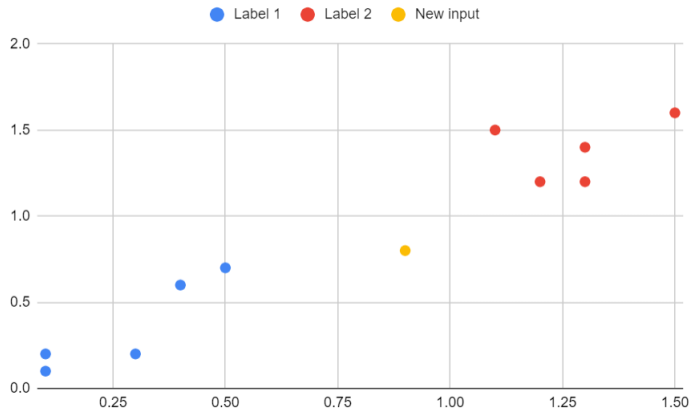

Figure 6.3: Visualisation of a simple KNN binary classifier with two input features.

The K-nearest neighbors classifier is visualized in figure 6.3. A number of labeled data points belonging to one of two classes are related to a new data point which is classified according to the method described above.

6.2.4 Inputs and outputs

Throughout this thesis the approach to the beam selection problem evolves and is iterated upon. For this reason the input features used by the model described above and their outputs vary from experiment to experiment. Generally speaking however, an array of input values is received by each model. The length of this array varies depending on the type of input data used. In classification tasks the output consists of a single value indicating the predicted label, while in other experiments several output values are predicted. The details of each model used and their input and output formats are described in the 'Experiment iterations' section.

6.3 The Keras API

All linear regression and DNN models used in this thesis were implemented using the Keras API. It provides a Python framework for interaction with TensorFlow, a commonly used machine learning library. The features of Keras most commonly used during this thesis are described below.

6.3.1 Normalization

Feature normalization was used in models used during testing. This means that all input features are translated into the range [0, 1], meaning all inputs will have the same scale. This is a helpful technique for some models, especially when the input data distribution is unknown [5].

6.3.2 Dense layer

The Dense layer is the Keras implementation of a regular connected neural network layer. Hidden layer and output layers are implemented using this feature in this thesis. Two important arguments required by the dense layer is the number of nodes within the layer and which activation function should be applied.

6.3.3 The Adam optimizer

The optimizer is the algorithm used to adjust the weights of a neural network during the training process. There are several algorithms that can be used for this purpose, however one of the most commonly used is the Adam optimizer. The Adam optimization algorithm uses stochastic gradient descent to arrive at an ideal set of weights [4].

Learning rate

An important argument used by the optimizer is the learning rate. This parameter governs the rate at which the model updates weights during the training process. Should this value be too small the model may not update weights properly and never achieve a high prediction accuracy. Inversely, should the value be to high the model may over correct resulting is accuracy swings during training.

6.4 SciKit Learn

SciKit learn is a Python machine learning library which was used to implement the K-nearest neighbors algorithm during this thesis. The KNN classifier can be easily implemented using the preexisting KNeighborsClassifier model where the value of k can be changed as desired.

CHAPTER 7

Experiment iterations

Over the course of this thesis a number of different approaches to the beam selection problem were tested. This was done in order to answer the second research question regarding how to make predictions. Included in this is testing the machine learning models described in the previous chapter in order to answer the third question. Question 1 is answered by evaluating the results of these tests. Designing the experiments, conducting them and evaluating their results represents the bulk of the work done in this thesis.

On several occasions the result of an experiment would reveal flaws in the methodology used, meaning adjustments often had to be made in order to get a useful result.

In this section the different approaches that were used over the course of the thesis are described. Input data and labeling are described and the machine learning models used to make various predictions are explained.

The results of these experiments will be presented in the "Results and Discussion" section. Discussion of the results and potential issues with each approach are also found under "Results and Discussion".

7.1 Experiment 1: SRS Channel Estimates

In the initial experiment the SRS Channel Estimates from the AndreAroundGarage.log file were used to generate the input data for the machine learning model. For each of the 64 beams in layer 0 of each channel estimate group the beam power was calculated using the method described above. These values were also plotted using a bar chart to visualize how beam power varies over the entire range. The indices of the amplitude peaks for each group

in the dataset were also plotted to evaluate the representativeness of the collected data. An array containing the beam power values for any given group of channel estimation data was used as input for the model.

In order to label this data two predefined beam regions were used, creating a binary classification problem. From the 64 beams these fixed regions were created, contiguously within the beam array. This mean that the first 32 values of the beam array make up the first region, while the rest makes up the second. These regions were labeled as '0' and '1' respectively

7.1.1 Machine Learning Models

Linear regression

A linear regression model was selected as a starting point for this experiment. As previously mentioned, this model attempts to find a linear relationship between the input variables, in this case the beam power values, and the index of the best beam region.

Layer	Type	Nodes	Activation function
1	Normalization	-	-
2	Dense	1	Linear

Table 7.1: The structure of the linear regression model used in the first experiment.

The linear regression model was implemented using the Keras API and was structured as shown in table 7.1. An initial normalization layer was used to independently normalize the input features. Secondly, a 'Dense'-layer containing a single unit in order to generate one output value was added. Here the linear activation function is applied in order to produce a result.

Variable	Value
Learning rate	0.0001
Batch size	100
Nbr of epochs	100
Loss function	mean_absolute_error

Table 7.2: The parameters used when training the linear regression model.

The parameters used to train the model were as shown in table 7.2. A 'mean absolute error'-loss function was used for evaluating the prediction accuracy of the linear regression model. Different values for learning rate, batch size and number of epochs were tried and evaluated in all experiments to determine which give the best prediction accuracy. A learning rate of 0.0001, batch size of 100 and fitting the model over 100 epochs gave the best result in the end and these values were thus chosen for this experiment.

Binary classification using a DNN

The second model used in the initial experiment was a Deep Neural Network (DNN). As described in the 'Machine Learning Models' section, a Deep Neural Network is created by placing several hidden layers between the inputs and the output of the model. Each of these layers stores and evaluates the significance of its different inputs. This information is then used as input for the next hidden layer. Each layer consists of several nodes, each of which takes into account the weighted outputs of the previous layer and generates outputs using an activation function.

Layer	Type	Nodes	Activation function
1	Normalization	-	-
2	Dense	64	ReLU
3	Dense	64	ReLU
4	Dense	1	Sigmoid

Table 7.3: The structure of the DNN model used in the first experiment.

Like the linear regression model, the DNN-model was implemented using the Keras API. Table 7.3 shows the structure. A normalization layer is again initially used for feature normalization. Second, two hidden layers make up the neural network. Each of these has 64 inputs and contains 64 nodes where a Rectified Linear activation function (ReLU) is applied. These layers are implemented using the Dense-layers present in the Keras API Finally, an output layer with a single node and a sigmoid activation function was used.

Variable	Value
Learning rate	0.0001
Batch size	100
Nbr of epochs	100
Loss function	binary_crossentropy

Table 7.4: The parameters used when training the DNN model.

Table 7.4 show the parameters used when fitting the model. A 'binary crossentropy'-loss function was used in this iteration of the experiment. A learning rate of 0.0001 and batch size of 100 were selected and the model fitting was run for 100 epochs.

Input data

The models in the first experiment were trained using the AndreAroundGarage.log file containing only SRS data. This means a total of 64 beam power values were used as input to predictr a single output class. After labeling the data was randomized and divided into training and testing data using an 80-20 split.

7.2 Experiment 2: Using PMI to predict top beams

The second experiment differs significantly from the initial one as beam power is no longer used as input for the machine learning model. Instead, PMI-data is used to predict where the beams with the highest power are located within the beam range.

The PMI-data is extracted from the log file which is structured as follows: A line containing the relevant PMI values is followed by a varying number of lines containing the SRS data. This is described in detail in the 'Data' section. The PMI-data corresponds to all following SRS data until a new PMI-line is encountered. The use of PMI-data as an input means that all models in the second experiment take an array of two index values as input while the number of outputs varies.

7.2.1 Predicting the entire range

During this experiment several ways of labeling the PMI data and predicting beam power peaks were tested. In the initial iteration, an attempt was made to predict the beam power for each of the 64 beams in the range. The locations of the desired number of top peaks could then potentially be identified from the predicted range. A list of 64 beam power values was therefore used to label the corresponding PMI data.

7.2.2 Predicting the index of beam power peaks

In the second iteration of this experiment the PMI data was instead used to predict where the beam power peaks would be located within the range of beams. This means that the model predicts the indices of the top beams instead of predicting the actual beam power values. The input PMI data used to train the model was therefore labeled using a list of indices, sorted in numerical order.

7.2.3 Machine learning models

In the second experiment a Deep Neural Network (DNN) model was again used. The principle of this model was the same as in the first experiment, creating several hidden layers to evaluate features and improve prediction accuracy. Each iteration of the experiment used somewhat different models.

Predicting the entire range

When predicting an entire range of beam power values one output is needed for each index, meaning that the first iteration of the DNN model has 64 outputs as seen in table 7.5. As the PMI data consists of only two values this is the length of the input vector.

Feature normalization is again used to make sure all features stay withing the same range. This is followed by two hidden layers, implemented using the

Layer	Type	Nodes	Activation function
1	Normalization	-	-
2	Dense	32	ReLU
3	Dense	64	ReLU
4	Dense	64	Sigmoid

Table 7.5: The structure of the DNN model used to predict the entire beam range in the second experiment.

Dense-layer in Keras. These consist of 32 and 64 nodes respectively and use a Rectified Linear activation function (ReLU). Finally, an output Dense-layer is used with 64 nodes and a sigmoid activation function.

7.2.4 Predicting the index of beam power peaks

Layer	Type	Nodes	Activation function
1	Normalization	-	-
2	Dense	4-8	ReLU
3	Dense	4-8	ReLU
4	Dense	4-8	Sigmoid

Table 7.6: The structure of the DNN model used to predict the indices of the beam power peaks in the second experiment.

When predicting the indices of beam power peaks instead of the entire range the model changes somewhat. Its structure is shown in table 7.6. The output vector only needs to have the length of the desired number of predicted peak locations. Output lengths of 8 and 4 were tested in this iteration of the experiment.

The model is similar to before. A normalization layer is followed by two hidden layers consisting of the same number of nodes as the output vector length (8 or 4) using a ReLU activation function. An output layer consisting of the desired number of output nodes with a sigmoid activation function is finally applied.

Variable	Value
Learning rate	0.001
Batch size	10
Nbr of epochs	100
Loss function	categorical_crossentropy

Table 7.7: The parameters used when training the DNN model in the second experiment.

Several parameter combinations were tested when building this model, but the best accuracy was yielded using the values in table 7.7. Both of the DNN-models use a learning rate of 0.001, batch size of 10 and are both trained over 100 epochs. A categorical cross entropy loss function was used as it is a better choice for cases where several values need to be predicted.

7.3 Experiment 3: Using PMI to predict one of two static regions

Experiment 3 has a similar objective to the first. Instead of trying to make the more complex predictions from experiment two, the problem is reduced to one of binary classification. The entire 64 beam range is again split into two predefined sections with the first 32 beams, or one polarization, as the first and the final 32 as the second. The objective is to predict, for a given PMI input pair, which of the two beam regions should be selected.

The input PMI data is labeled with the index of the appropriate region, either a 1 or a 0. Which region is best suited is determined using the SRS data also present in the log. The beam power is calculated for each of the 64 beams in the range and then the amplitudes of the top 8 highest power beams for each region are summarized. Whichever region has the highest sum is selected as the label. The models in the third experiment take the same two inputs as in the previous experiment wile only predicting a single output class.

7.3.1 Machine learning model

Linear regression

First, a linear regression model was tested for the case when PMI data is used for binary classification. This is to make a similar comparison to experiment 1 where two models were tested.

Layer	Type	Nodes	Activation function
1	Normalization	-	-
2	Dense	1	Linear

Table 7.8: The structure of the linear regression model used to predict predefined regions in the third experiment.

The original linear regression model is once again used, however with only two inputs instead of 64. A normalization of the input features is followed by a single node which applies a linear activation function, as seen in table 7.8.

This linear regression model also uses the same parameters as before (see table 7.9). A learning rate of 0.0001, batch size of 100 are used and the model is trained over 100 epochs. To evaluate results and improve the model a mean absolute error loss function is used.

Variable	Value
Learning rate	0.0001
Batch size	100
Nbr of epochs	100
Loss function	mean_absolute_error

Table 7.9: The parameters used when training the linear regression model in the third experiment.

Deep Neural Network

A Deep Neural Network (DNN) model similar to the ones from the previous experiments is also used in the third experiment.

Layer	Type	Nodes	Activation function
1	Normalization	-	-
2	Dense	2	ReLU
3	Dense	2	ReLU
4	Dense	1	Sigmoid

Table 7.10: The structure of the DNN model used to predict predefined regions in the third experiment.

As the intended prediction only consists of a single index only one output value is required (see table 7.10). All input features are again normalized using an initial normalization layer. This is followed by two hidden layers with two nodes each and ReLU activation functions. The final Dense layer applies a sigmoid activation function and produces a single output.

Variable	Value
Learning rate	0.001
Batch size	10
Nbr of epochs	100
Loss function	mean_absolute_error

Table 7.11: The parameters used when training the DNN model in the third experiment.

The best prediction accuracy for this model was achieved with a learning rate of 0.001 and a batch size of 100, as shown in table 7.11. Model fitting occurred over 100 epochs. In this experiment a mean absolute error loss function was again used.

7.4 Experiment 4: Using PMI to predict one of four static regions

In the final experiment attempts are made to use the model from the previous scenario in a more useful context. Utilizing only one fourth of the beam array could potentially lead to even greater energy savings compared to if half of it is used. The goal of this experiment is therefore to investigate whether the DNN model can predict the appropriate beam region when the beam array is split in four instead of two.

In a similar manner to before the PMI input data is labeled with a beam region index, however these are now in the range of 0-3. Labeling is done in the same way as in the previous experiment with the top 8 beams of each region being summarized and compared to determine the label. The number of inputs and outputs of each ML-model remains the same as in the third experiment.

7.4.1 Machine learning models

Deep Neural Network

Layer	Type	Nodes	Activation function
1	Normalization	-	-
2	Dense	2	ReLU
3	Dense	2	ReLU
4	Dense	1	Sigmoid

Table 7.12: The structure of the DNN model used to predict predefined regions in the fourth experiment.

The Deep Neural Network (DNN) model present in experiment 3 was applied to this problem as well. Its structure remains the same, consisting of a normalization layer, two hidden layers with ReLU activation functions, and a final output with sigmoid activation (see table 7.12).

Variable	Value
Learning rate	0.001
Batch size	10
Nbr of epochs	100
Loss function	mean_absolute_error

Table 7.13: The parameters used when training the DNN model in the fourth experiment.

The same parameters were also used for this experiment, as shown in table 7.13. A learning rate of 0.001, batch size of 100 and 100 epochs gave the best accuracy. The loss function used was 'mean_absolute_error'.

K-nearest neighbors

For comparison of the results achieved using the DNN model another classification method was also tried in the fourth experiment, namely the K-nearest neighbors algorithm.

K-nearest neighbors classification makes a prediction by looking at the k labeled data points that are the shortest distance from the input. The most common label among these neighbors is selected as the class for the input data.

Parameter	Value
k	5, 10, 100
weights	uniform
algorithm	auto

Table 7.14: The parameters used for the K-nearest neighbors classifier in the fourth experiment.

This algorithm was implemented using the KNeighborsClassifier module from the Scikit-learn python library. Three different values for k were tested; 5, 10 and 100 (see table 7.14). All other parameters were set to their default values. Most importantly, uniform weights were used, meaning all neighboring data point were weighted equally. The algorithm used to compute the nearest neighbors was set to 'auto', meaning that a search algorithm was selected based on the inputs given. The KNN classifier works of the same dataset as the DNN model, meaning it takes two input features and produces a single output.

Input data

Both models were trained using the PMI_SRS_around_watertower.log file containing both SRS and PMI data. The input data was randomized and divided between training and testing using an 80-20 split. Predictions were made on the randomized testing data.

An attempt was also made to use the PMI values from the PMI_SRS_garage_top.log file as testing input data while still training the model on the data from PMI_SRS_around_watertower.log. This was done to determine whether the model would be accurate in a scenario where the input data is not represented in the training dataset.

CHAPTER 8

Results and Discussion

In this chapter the results of the experiments described in the previous section are presented. The results are listed in the order that they were conducted and are then discussed. Potential flaws in experiments are identified and examined and the the modifications made in between experiments are explained.

$$accuracy = \frac{nbrofcorrectpredictions}{totalpredictions}$$

The metric used to evaluate the viability of a machine learning model is the prediction accuracy. The accuracy score is implemented in Keras and is calculated by dividing the number of correct predictions by the total number of testing data points. A prediction is correct if the predicted label matches the label assigned to the PMI data point during the labeling process.

Another metric used to evaluate each model is loss. Loss is an indicator for how far away from their predefined labels the predictions of the model are on average. A model with a high loss value is therefore less accurate than one with a low loss. The loss function used to calculate this value is the same one defined when compiling the model.

Both of these metrics are obtained by running the evaluate() function from Keras after the model has been trained for the specified number of epochs.

The results of the tests conducted during this thesis are presented in tables containing the values for accuracy and loss achieved below. As input data is always randomized before being split into training and testing datasets there will always be some variation in prediction accuracy when training the model several times. For this reason models were each trained ten times and the deviations in accuracy and loss are included in the tables as well.

8.1 Plotting beam power

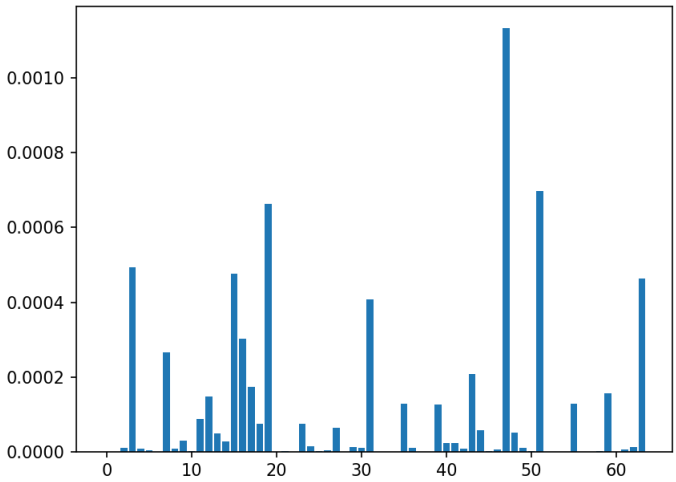

Figure 8.1: Beam power distribution for group nbr 68 at timestamp 2022-07-20 08:25:21.448061 in AndreAroundGarage.log.

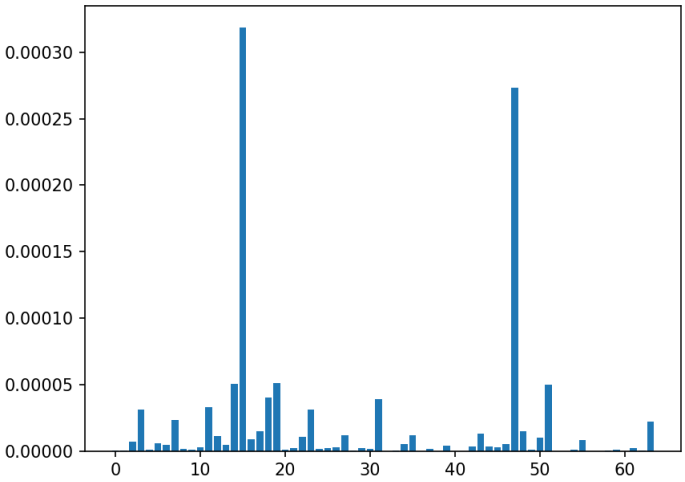

Figure 8.2: Beam power distribution for group nbr 119 at timestamp 2022-07-20 08:31:37.212065 in AndreAroundGarage.log.

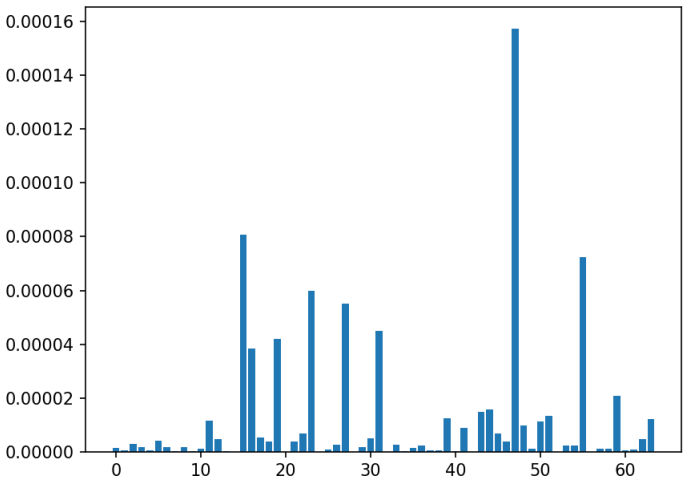

Figure 8.3: Beam power distribution for group nbr 17 at timestamp 2022-07-20 09:02:02.360573 in AndreAroundGarage.log.

To visualize the the beam power range for a group of channel estimation data a number of randomly selected groupss from the AndreAroundGarage.log file were plotted (see figures 8.1, 8.2 and 8.3). A total of three groups from three different timestamps were selected. These have the timestamps and group numbers listed above.

In these example plots there are two clear main peaks within the range the located in around the same two spots. This cpould potentially suggest that the collected data doesn't cover enough use cases. A dataset where peaks are spread across the range could potentially be more useful. To analyze this potential issue the indices of all the largest beam power peaks were extracted and their respective number of occurrences were counted. These counts were then plotted.

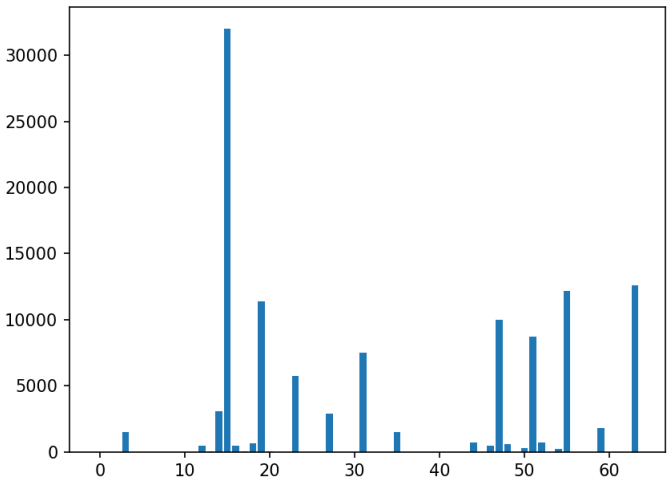

Figure 8.4: Distribution of the highest amplitude peak for each group in AndreAroundGarage.log.

After parsing the data in the log file and calculating the beam power values the peak locations were counted.

As we can see in figure 8.4 there is a clear pattern for where within the range the amplitude peaks are located. Because of this pattern this particular data set would not be optimal for non-binary classification. Should the range be split into four regions two of them would not be represented in the training data. This is why only binary classification is attempted in the first experiment. In real world use cases with several UEs moving in different patterns would create a more diverse data set. This is the reason why new, more representative data is collected for the latter experiments.

8.2 Experiment 1: SRS Channel Estimates

8.2.1 Linear regression model

Metric	Value
Accuracy	0.86 ± 0.01
Loss	0.16 ± 0.01

Table 8.1: Accuracy and loss achieved using a linear regression model with data from AndreAroundGarage.log as input.

When making predictions using the linear regression model in the first experiment an accuracy of about 86% is achieved with an average loss of about 0.16 (see table 8.1).

8.2.2 Binary classification using a DNN

Metric	Value
Accuracy	0.95 ± 0.01
Loss	0.16 ± 0.01

Table 8.2: Accuracy and loss achieved using a DNN model with data from AndreAroundGarage.log as input.

In the case where a DNN-model model is used a higher accuracy is achieved, as shown in table 8.2. When the Deep Neural Network is used 95% of predictions are accurate, with an average loss of 0.05.

The DNN-model is clearly the better choice when the goal is to make accurate predictions, although the linear regression model is still able to make predictions with a relatively high degree of certainty.

The high performance achieved in this experiment is not unexpected. Performing binary classification with a comparatively large number of inputs is a relatively simple task. The occurrence of some incorrect predictions is likely caused by the input data. As seen in figures (Insert reference) the two amplitude peaks in the collected data are often of similar size and shape. This is a result of the cross-polarized nature of the antenna array. Because of the similarity between the two peaks it can be difficult for a model to determine which classification should be applied to a given input. This is especially the case for the linear regression model, which is the simpler of the two. Therefore, prediction accuracy could potentially be improved by defining regions in a different manner.

8.2.3 Issues with the experiment

After analyzing the results of this first experiment a number of issues were identified which lead to this avenue of testing being abandoned.

There is not necessarily anything wrong with the results themselves. In fact, as was demonstrated above, the accuracy achieved was very good. The major issue is that the methodology used, and by extension the design of the machine learning model, would not be especially useful in a real world scenario.

Compared to PMI-information, channel estimation data is recieved much more frequently. This can be seen clearly in the SRS_and_PMI.log file. There are commonly hundreds of estimation data point recorded between updates of the PMI indices. This means that an approach where a region is predicted upon each new channel estimation message would be very costly. This issue could be remedied by only making predictions periodically, at a predefined interval.

The usefulness of training the machine learning model using SRS-data is however still questionable. Calculating the optimal region directly would likely be an easier approach. If a machine learning model is used, the entire range of beam power values need to be calculated and fed to the model. Only then can

a prediction be made. As we go through the trouble of calculating the entire beam power range, it would be easier to just compare these values directly and select a region based on the result. This makes any machine learning model redundant.

Whether selecting a region directly in the manner described above or predicting using the ML-model, the process of calculating beam power for the entire range is a costly one. Even if predictions are only made periodically, using resources for this purpose is wasteful.

Another issue with using channel estimation data is that it does not include information about the physical location of the UE in the same way that PMI-data would. While SRS contains a lot of other information that could potentially be relevant to making a prediction, the location of the UE is highly relevant to where within the beam range amplitude peaks will be located. This is yet another reason why the decision was made to make predictions based on PMI instead of channel estimates going forward.

8.3 Experiment 2: Using PMI to predict top beams

8.3.1 Plotting beam power peaks

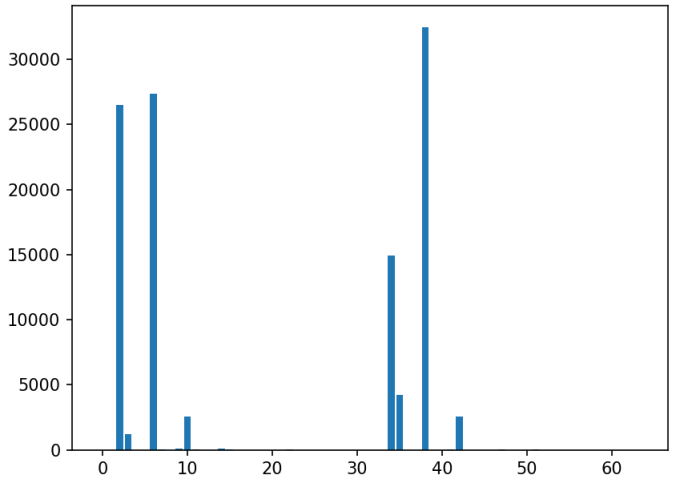

Figure 8.5: Distribution of highest amplitude peak for each group in SRS_and_PMI.log.

The distribution of amplitude peaks in SRS_and_PMI.log (see figure 8.5) is similar to that of the first log file (see figure 8.4). The entire range is not well represented, which may have an impact on the usefulness of the model in the real world where a more diverse range of scenarios will be encountered. In later experiments more representative log files are used.

8.3.2 Predicting the entire range

Metric	Value
Accuracy	0.0164 ± 0.002
Loss	143 ± 0.01

Table 8.3: Accuracy and loss achieved using a DNN model with data from SRS_and_PMI.log as input.

The attempt to predict the entire range of beam power values using a DNN model resulted in a prediction accuracy of 0.0164 and an average loss of 143, as shown in table 8.3. These values were achieved when the model was compiled using a categorical crossentropy loss function.

Obviously this result is very poor. The main reason for such a low accuracy is most likely that the model is underfitted. One scenario in which underfitting can occur is when the model has too few input features. This leads to the model being unable to discern trends within the data causing large errors. It is a very difficult task to predict 64 values based on just two inputs, and for that reason accurate predictions can not be made.

There is no reason to attempt classification based on the predicted range. Assigning a label based on the peaks given by the model would be fruitless as the results are just too inaccurate.

8.3.3 Predicting the index of beam power peaks

To address the issues encountered when trying to predict the entire range the number of output from the model had to be reduced. Predicting a small number of indices for the highest amplitude beams seemed like a reasonable solution. This way the number of values to predict could be reduced while still keeping the most relevant data. Regions could then potentially be assigned based on the locations of the predicted peaks.

Metric	Value
Accuracy	$6.2099e^{-}04 \pm 1e^{-}04$
Loss	1426 ± 100

Table 8.4: Accuracy and loss achieved using a DNN model with data from SRS_and_PMI.log as input when trying to predict the top 8 beam indices.

Metric	Value
Accuracy	0.36 ± 0.03
Loss	669 ± 50

Table 8.5: Accuracy and loss achieved using a DNN model with data from SRS_and_PMI.log as input when trying to predict the top 4 beam indices.

When attempting to predict the 8 beam indices corresponding to the 8 highest amplitude beams an accuracy of $6.2099e^{-}04$ was achieved with a loss of 1426 (see table 8.4). The same experiment yeilded an accuracy of 0.36 and loss of 669 when predicting only the top 4 values instead (see table 8.5).

The results of this experiment are clearly not very accurate. Making predictions using this model in a real world scenario would not be especially useful. The problem of the model being underfitted persists to an extent which has an impact on performance. Another issue may be that a prediction will be considered incorrect if an index is predicted close to the correct value, but not exactly correct. In other words small variations in predictions can cause inaccuracy to decrease. As can be seen in figure 8.5, amplitude peaks are not evenly distributed for this particular log file. The peaks are concentrated on a small number of indices with adjacent indices being underrepresented. This means that making an accurate prediction as to the location of amplitude peaks is extra important as an inaccurate prediction will be less useful in a real world scenario.

Logically, underfitting would be less of an issue with a smaller number of outputs. Evidence of this can be seen in table 8.5 where accuracy improves significantly when going from 8 to 4 predicted indices. The issue of inaccuracy caused by small deviations in predictions remains however, and the accuracy achieved is still a lot lower than desired. This means that this iteration of the model will also be ineffective when applied in the real world.

8.3.4 Issues with the experiment

As mentioned in the previous section underfitting and inaccuracy caused by small variations in predictions are large problems with the models in experiment 2.

In addition to this a problem with the experiment is that when the log file is parsed the amplitude peak data and corresponding PMI values are stored using the JSON file format. A JSON file behaves in a similar manner to a python dictionary in that it stores data in key-value pairs. This means that only one example of a specific key can exist in a given file. The current log parsing implementation pairs a label consisting of 4 or 8 values with a PMI-value pair and inserts them into the JSON file. As a result of this a label of beam indices will only appear once in the training data for the model. This causes large issues as large amounts of repeating data will be excluded, impacting the accuracy of predictions. This issue is remedied in experiments 3 and 4.

8.4 Experiment 3: Using PMI to predict one of two static regions

In the third experiment the issues encountered in the second experiment are addressed. By simplifying the problem to a binary classification one, the problem of inaccuracies caused by small variations in predicted indices is resolved. Reducing the number of outputs to just one also helps address the problem of model underfitting.

In addition to this, the issues with the log parsing process present in experiment 2 have been fixed. This was done by saving SRS and PMI data collected from the log file in a .txt file instead of a JSON file. This way label-feature pairs can have the same key and still be include in the training data. Repeating data is now also included which affects the weighting process during model training to make predictions more accurate.

8.4.1 Plotting beam power peaks

For the third experiment the SRS_and_PMI.log file was initially used. As was discovered in the previous experiment (see figure 8.5) the beam power peaks in this log are not equally distributed across the range. This is fine for binary classification, but in the fourth experiment more representative data is needed. In addition to this the amount of data in the log is relatively small, which can have an impact on prediction accuracy. For this reason, another log file, PMI_SRS_around_watertower was collected. In this experiment both logs were tested. The peak amplitude distribution for the entire log was plotted.

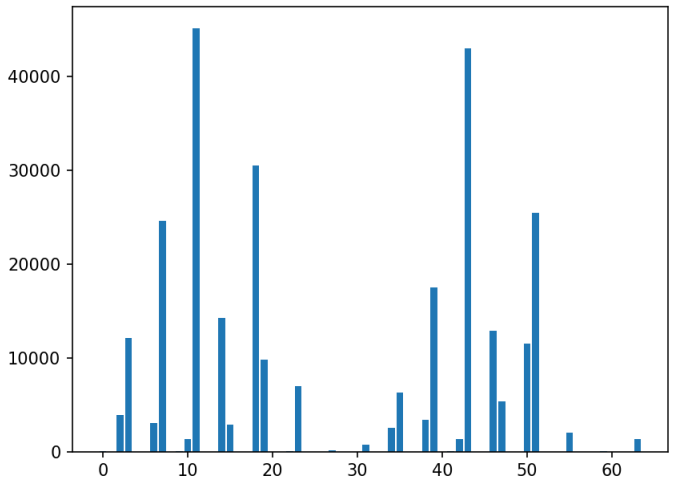

Figure 8.6: Distribution of highest amplitude peak for each group in PMI_SRS_around_watertower.log.

As seen in figure 8.6 the amplitude peak distribution for the new log file is much more even compared to the on seen in figure 8.5. There is still a clear trend in the locations of the peaks, however more indices are represented.

8.4.2 Filtering of data

To improve upon the accuracy that was initially achieved in this experiment tests were also made using filtered data. After parsing each log file there are many examples of a PMI value pair being labeled with different labels depending on the distribution of beam power across the beam range. This can have a negative effect on prediction accuracy as features having several labels will "confuse" the model during training. For this reason the input data was filtered by only including data labeled with the dominant label. In other words, if there are 400 examples of the PMI pair '14,4' being labeled '1' and 200 examples of it being labeled '0' only the data points labeled '1' will be included. Both unfiltered and filtered log files were tested during this experiment.

8.4.3 Linear regression

A linear regression model was first tested for the sake of comparison with the DNN model that was subsequently used. Both the old SRS_and_PMI.log and new PMI_SRS_around_watertower.log files were tested, unfiltered and filtered.

Unfiltered data

Metric	Value
Accuracy	0.522 ± 0.03
Loss	0.468 ± 0.03

Table 8.6: Accuracy and loss achieved using a linear regression model with unfiltered data from SRS_and_PMI.log as input.

Metric	Value
Accuracy	0.531 ± 0.02
Loss	0.459 ± 0.02

Table 8.7: Accuracy and loss achieved using a linear regression model with unfiltered data from PMI_SRS_around_watertower.log as input.

Using an unfiltered dataset the accuracy achieved is not very high. With the two log files only 52% and 53% of predictions were correct respectively as seen in tables 8.6 and 8.7. This is not especially surprising as having inputs labeled differently will cause accuracy to decrease.

Filtered data

Metric	Value
Accuracy	0.709 ± 0.05
Loss	0.291 ± 0.05

Table 8.8: Accuracy and loss achieved using a linear regression model with filtered data from SRS_and_PMI.log as input.

Metric	Value
Accuracy	0.561 ± 0.02
Loss	0.447 ± 0.03

Table 8.9: Accuracy and loss achieved using a linear regression model with filtered data from PMI_SRS_around_watertower.log as input.

In the case of using the linear regression model on filtered datasets a data outlier can be seen. When using the smaller log file, accuracy increases to around 71% as seen in table 8.8, however when running the same model on the larger dataset accuracy drops to just 56% (see table 8.9). This is unexpected as a larger training dataset would generally coincide with better predictions being made. It is not entirely clear as to why this outlier appears. One potential cause may be that the more equally distributed amplitude peaks in the PMI_SRS_around_watertower.log makes it more difficult for a simple linear model to determine a linear relationship between the PMI data and label.

8.4.4 Deep Neural Network

Unfiltered data

Metric	Value
Accuracy	0.547 ± 0.02
Loss	0.454 ± 0.03

Table 8.10: Accuracy and loss achieved using a DNN model with unfiltered data from SRS_and_PMI.log as input.

Metric	Value
Accuracy	0.558 ± 0.03
Loss	0.441 ± 0.03

Table 8.11: Accuracy and loss achieved using a DNN model with unfiltered data from PMI_SRS_around_watertower.log as input.

Using the unfiltered data as input for the deep neural network model shows a similar result as using it for the linear model. As seen in tables 8.10 and 8.11 accuracies of 55% and 56% are achieved respectively for the two log files. The DNN gives a slightly better result than the linear model in this case, however the issues caused by not filtering the data persist leading to decreased accuracy.

Filtered data

Metric	Value
Accuracy	0.751 ± 0.05
Loss	0.264 ± 0.04

Table 8.12: Accuracy and loss achieved using a DNN model with filtered data from SRS_and_PMI.log as input.

Metric	Value
Accuracy	0.874 ± 0.02
Loss	0.132 ± 0.03

Table 8.13: Accuracy and loss achieved using a DNN model with filtered data from PMI_SRS_around_watertower.log as input.

Using the deep neural network after filtering the data provides the most promising result yet. Accuracies of 75% and 87% are achieved respectively for the two log files (see tables 8.12 and 8.13). The DNN is better able to find and quantify a relationship between PMI values and region index compared to the linear model. Using a larger input dataset also improves accuracy in this case, which is in line with what would be expected. The range of achieved prediction accuracies is also tighter when using the larger log file, which is indicative of a higher level of confidence and less variability.

8.5 Experiment 4: Using PMI to predict one of four static regions

Dividing the antenna array into more than two regions could potentially be beneficial and yield greater efficiency as an even smaller number of antennas

could be activated at any given time. For this reason the fourth experiment introduces two more classes to the classification problem described in experiment 3, dividing the beam range into 4 regions.

To evaluate and compare the results achieved using the DNN model a K-nearest neighbors algorithm was also tried for classification of the data.

Models were tested using the PMI_SRS_around_watertower.log file used in the previous experiment. Using this file is what enables non-binary classification as its amplitude peaks are more evenly spread across the beam range. This means that when the range is split into four regions they will all be represented in the dataset. All data is filtered.

In addition to testing each model by splitting the input dataset into testing and training data another approach was also tried. By using a separate dataset for testing the behavior of the model when receiving data not present in the training dataset can be evaluated. To produce this second input dataset the PMI_SRS_garage_top.log file was used.

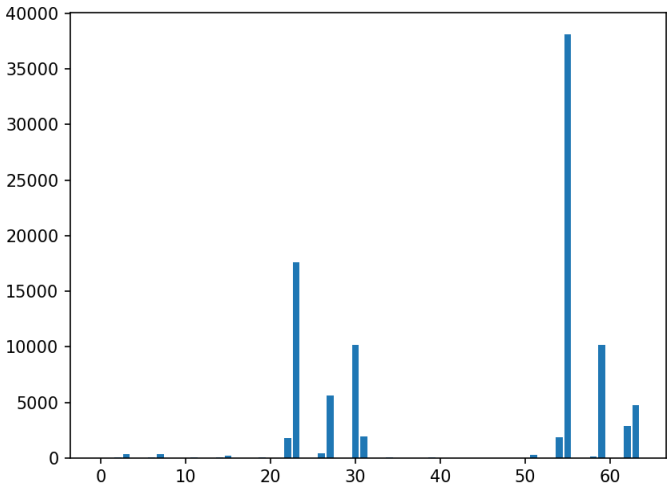

Figure 8.7: Distribution of highest amplitude peak for each group in PMI_SRS_garage_top.log.

The distribution in figure 8.7 shows that the beam power peaks in this log file are less evenly distributed compared to the previous log file, shown in figure 8.6. This is not especially significant as the log is only being used for testing and to determine the usefulness of the model in a real world scenario.

8.5.1 Deep Neural Network

Metric	Value
Accuracy	0.369 ± 0.02
Loss	0.756 ± 0.02

Table 8.14: Accuracy and loss achieved trying to predict 1 of 4 regions using a DNN model with filtered data from PMI_SRS_around_watertower.log as input.

After training the DNN model for 100 epochs an accuracy of 0.369 and loss of 0.756 were achieved as shown in table 8.14. This is obviously much lower than for the binary classification case, however the result is still an improvement on random selection of regions.

Metric	Value
Accuracy	0.292 ± 0.02
Loss	1.409 ± 0.03

Table 8.15: Accuracy achieved trying to predict 1 of 4 regions using a DNN model with filtered data from PMI_SRS_garage_top.log as input.

As shown in table 8.15 and accuracy of 0.292 and loss of 1.409 were achieved when the testing dataset was swapped from the PMI_SRS_around_watertower.log file to the PMI_SRS_garage_top.log file. While this prediction accuracy is not as high as in the first test (see figure 8.14) it is still a promising result as it is within 8 percentage points of the original test. This suggests that it could be viable to use a DNN for classification, even when it is given inputs not present in the training data. Such scenarios are likely to occur in a real world use case.

The prediction accuracies of both these tests are nowhere close to an acceptable value. It may be necessary to make predictions on other input features in addition to the PMI values. This is further discussed in the 'Future work' secrion.

8.5.2 K-nearest neighbors

To help evaluate the results of the DNN model a K-nearest neighbor approach was tried using the k-values 5, 10 and 100.

Metric	k = 5	k = 10	k = 100
Accuracy	0.999	0.998	0.987

Table 8.16: Accuracy and loss achieved trying to predict 1 of 4 regions using a DNN model with filtered data from PMI_SRS_around_watertower.log as input.

As table 8.16 shows, no matter the k-value chosen an accuracy close to 100% is achieved. This is due to the fact that the testing data largely contains the same data points as the training data. As the K-nearest neighbors algorithm classifies an input by looking at neighboring points, most of the neighbors will have the same value. This leads to a near perfect prediction, however this result is not especially useful in a real world scenario where you cannot count on the PMI data sent by a given UE being present in the training data. To investigate this further the PMI_SRS_garage_top.log file was parsed and tried as input.

Metric	k = 5	k = 10	k = 100
Accuracy	0.192	0.149	0.143

Table 8.17: Accuracy and loss achieved trying to predict 1 of 4 regions using a DNN model with filtered data from PMI_SRS_garage_top.log as input.

As seen in table 8.17 the accuracies achieved when using the data from another log file as testing data is much lower. 0.192, 0.149 and 0.143 for k-values 5, 10 and 100 respectively. This indicates that many of the PMI pairs present in the first log are not present in this one. Therefore using the k-nearest neighbors method may not be a suitable approach when many PMI-pairs are not present in the training dataset.

When compared to the results achieved using a DNN, seen in table 8.15, the DNN model appears more favorable. The deep neural network makes significantly more accurate predictions based on previously unseen input data. For this reason the DNN model is likely more appropriate for non-binary classification.

CHAPTER 9

Conclusions and Future Work

In this chapter any conclusions that can be drawn from the experiments above will be presented and discussed. In addition to this potential future work on this topic will be examined.

9.1 Conclusions

The overarching aim of this thesis was to investigate the possibility of using machine learning techniques for the purpose of antenna selection. Approaching this problem by making a prediction in the beam space was chosen as a good starting method. Through the process of investigating this problem several approaches were tested and their ability to accurately predict a region in the beam space were evaluated.

9.1.1 Q1: Is it possible to accurately predict a subset of beams to use using a machine learning model.

As shown, primarily in the third experiment, it is possible to accurately predict a predefined beam region when the problem is reduced to one of binary classification. A prediction accuracy of 87.4% was achieved when using the larger of the datasets as input to a DNN model.

9.1.2 Q2: How should predictions be made and based on what input data?

Of the methods tested the first two did not show much promise. Using beam power to predict a label which is also decided by calculating beam power is

wasteful. In the second approach using PMI data to predict the indices of the highest amplitude beams in the beam range was attempted. Due to the model being underfitted when predicting a larger array of values based on just two inputs the second experiment did not produce a satisfactory accuracy rate.

Overall however, using PMI data to make predictions did show promise. In the final tests the problem was once again reduced to one of single label classification, meaning the prediction of predefined beam regions. In the third experiment binary classification based on PMI data was attempted and was shown to be a viable solution. The prediction accuracy when using a Deep Neural Network model was good.

Finally, classification when the beam range is split into four regions was also attempted. For this case the current model architecture was shown to be insufficient, with low prediction accuracies.

In the final experiment tests were also made using different datasets for training and testing. Giving the model testing data that includes inputs it has not yet encountered during training did have some impact on accuracy. This impact was however not great, and the model was still able to make predictions with better accuracy than randomly assigning labels. This is an indication that using a DNN to predict based on PMI data may be part of a viable solution in the non-binary classification case.

In conclusion, of the approaches tested, using a DNN model to make binary predictions based on PMI input data yielded the highest accuracies throughout the testing process.

9.1.3 Q3: Which machine learning model yields the highest prediction accuracy?

Although testing many different models did not end up being the main focus of the thesis conclusions can still be drawn from the tests that were conducted. The DNN models used consistently yielded higher prediction accuracies than linear regression as they were better able to capture the relationships between inputs and outputs.

The overall conclusion of the thesis is that predicting a predefined antenna region based on PMI-data is a viable technique. If there are more than two predefined regions other inputs may be needed to make an accurate prediction. A Deep Neural Network appears to be viable model for classification in the experiments conducted during this thesis.

9.2 Future work

The results of this thesis show that there is a lot of potential future work to do to develop this concept further.

While using the PMI data for binary classification did show promise and produced accurate results, this was not the case when trying to predict one of four beam regions. It is likely that by only using PMI data the model may not

be able to yield precise enough predictions. For this reason other inputs may be needed to make non-binary classification viable. During the thesis using the angle between the base station and UE as a parameter was discussed. Whether this or other inputs are feasible options is a matter that could be evaluated and tested in the future.

Adding more inputs to the model may also make it less prone to underfitting. This could potentially open up the possibility of predicting the indices of the highest amplitude beams, as was attempted in the second experiment. Predicting which beams to include in a region dynamically could potentially allow for greater efficiency, as it would be ensured that only the highest power beams are selected. This is another avenue that could be explored in the future. If classification is to be done using static regions other definitions than those used during this thesis could be experimented with.

Related to this, labeling data in alternate ways could also be explored. As the locations of beam power peaks can vary somewhat with time, taking into account several groups of channel estimate values when labeling input data may produce more representative and consistent results. Looking at several points in time when making predictions is also an avenue that could be explored for the same reasons.

9.2.1 Other ML models

In addition to adding more inputs other machine learning models may also be tested. For the purpose of focusing the thesis on testing several different approaches to the region prediction problem, and finding one that works well, only two main machine learning models were tested. There are however many other machine learning models that could be tested and may potentially provide even more accurate results. The decision tree is a model that was discussed and could be tried in the future.

When testing future models and deciding how to use these for antenna selection there are also several aspects which will need to be investigated further. At which interval predictions should be made is an important question that will need resolving if the system is to be used in a real world application. Different potential model inputs may be updated at different intervals, meaning that making predictions each time some inputs are updated may be too costly. In addition to this, the time it takes to make a prediction will depend on the complexity of the model. These two factors and how they affect the broader system must be investigated taken and into account when making real time predictions.

9.2.2 Antenna selection

There is also work to be done in order to achieve the main goal of solving the antenna selection problem. The beam predictions currently made exist in the beam space. In order to solve antenna selection the algorithm will have to be extended in such a way that beam space predictions can be converted to the

antenna space. This problem will require future investigation and implementation. Any solution will also have to be tested in a simulation or real world environment, as was the original intention of this thesis.

References

[1] Pragati Baheti. Activation functions in neural networks, 2021. Last accessed 18 August 2023.

[2] Jason Brownlee. How to configure the number of layers and nodes in a neural network, 2019. Last accessed 18 August 2023.

[3] Jason Brownlee. A gentle introduction to cross-entropy for machine learning, 2020. Last accessed 18 August 2023.

[4] Jason Brownlee. Gentle introduction to the adam optimization algorithm for deep learning, 2021. Last accessed 19 August 2023.

[5] deepchecks. Normalization in machine learning. Last accessed 19 August 2023.

[6] J. Peisa E. Dahlman, S. Parkvall. 5g evolution and beyond. *IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications*, 2019.

[7] Rosdiadee Nordin Ehab Ali, Mahamod Ismail and Nor Fadzilah Abdulah. Beamforming techniques for massive mimo systems in 5g: overview, classification, and trends for future research. *Frontiers of Information Technology & Electronic Engineering*, 18(6):753–772, 2017.

[8] Ericsson. Massive mimo for 5g networks. Last accessed 21 August 2023.

[9] Fredrik Tufvesson Erik G Larsson, Ove Edfors and Thomas L. Marzetta. Massive mimo for next generation wireless systems. *IEEE Communications Magazine*, 52(2):186–195, 2014.

[10] Yuan Gao and Thomas Kaiser. Antenna selection in massive mimo systems: Full-array selection or subarray selection? In *2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 1–5, 2016.

[11] Kyriaki Psara Harun Siljak and Anna Philippou. Distributed antenna selection for massive mimo using reversing petri nets. *IEEE Wireless Communications Letters*, 8(5), 2019.

[12] IBM. What is the k-nearest neighbors algorithm? Last accessed 19 August 2023.

[13] Vandit Jain. Everything you need to know about "activation functions" in deep learning models, 2019. Last accessed 18 August 2023.

[14] Meenu Khurana. Deep learning based low complexity joint antenna selection scheme for mimo vehicular adhoc networks. *Expert Systems with Applications*, 219, 2023.

[15] Sandhya Krishnan. How do determine the number of layers and neurons in the hidden layer?, 2021. Last accessed 18 August 2023.

[16] Vlastimil Martinek. Cross-entropy for classification, 2020. Last accessed 18 August 2023.

[17] Muhib Ur Rahman Sher Ali, Amir Haider Malik. Deep learning (dl) based joint resource allocation and rrh association in 5g-multi-tier networks. *ResearchGate*, 2021.

[18] Telecoma. Massive mimo - benefits and challenges, 2023. Last accessed 20 August 2023.

[19] Lina Yuan Tongcai Wu and Anran Zhou. Antenna selection technology research in massive mimo system. *2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 2022.

[20] Analytics Vidhya. Understanding loss function in deep learning, 2013. Last accessed 8 August 2023.

[21] Thang X. Vu, Lei Lei, Symeon Chatzinotas, and Björn Ottersten. Machine learning based antenna selection and power allocation in multi-user miso systems. In *2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, pages 1–6, 2019.

[22] Wentao Yu, Tianyu Wang, and Shaowei Wang. Multi-label learning based antenna selection in massive mimo systems. *IEEE Transactions on Vehicular Technology*, 70(7):7255–7260, 2021.

[23] Shida Zhong, Haogang Feng, Peichang Zhang, Jiajun Xu, Huancong Luo, Jihong Zhang, Tao Yuan, and Lei Huang. Deep learning based antenna selection for mimo sdr system. *Sensors*, 20(23), 2020.