Stellenbosch University

Masters Thesis

Machine Learning for Antenna Array Failure Analysis

Author:
Lydia de Lange

Supervisors:
Dr. DJ Ludick
Dr. TL Grobler

Report submitted in partial fulfilment of the requirements of the module Project (E) 448 for the degree *Masters in Engineering* in the *Department of Electrical and Electronic Engineering* at the *Stellenbosch University*

March 2020

Acknowledgements

I would like to express my sincere thanks and appreciation to the following people for the support they offered me with regard to the project:

- To my supervisors, Dr DJ Ludick and Dr TL Grobler, thank you for our weekly meetings, your enthusiasm and ideas, and giving me the opportunity to go to conferences and summer schools, where I could learn so much more in the fields of antennas, radio astronomy, and machine learning.
- To my family, thank you for your unwavering love and support.
- To my friends, thank you for being attentive and patient soundboards whenever I was trying to figure something out.
- To Wessel Croukamp and Wynand van Eeden, thank you for the fun learning experience we had when you helped me build the 16-element circular patch antenna array.
- To my parents and Anne Denniston, I cannot thank you enough for your help with editing my thesis. I could not have done this without you.

Plagiaatverklaring / Plagiarism Declaration

- Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
  Plagiarism is the use of ideas, material and other intellectual property of another person and presenting it as one's own.

- Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
  I agree that plagiarism is a punishable offence because it constitutes theft.

- Ek verstaan ook dat direkte vertalings plagiaat is.
  I understand that direct translations are also plagiarism.

- Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
  Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.

- Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
  I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously submitted it (in its entirety or in part) for grading in this module/assignment or another module/assignment.

Voorletters en van / Initials and surname: L de Lange

Datum / Date: March 2020

STELLENBOSCH UNIVERSITY

Abstract

Faculty of Engineering
Department of Electrical and Electronic Engineering

Masters in Engineering

Machine Learning for Antenna Array Failure Analysis

by Lydia de Lange

This work investigated the use of machine learning to detect failed elements in an antenna array. The aim was to identify a trustworthy means of early detection and isolation of faulty elements to improve the reliability of measured data. Previous work has shown that it is theoretically possible to identify failed elements from the far-field radiation pattern, using machine-learning algorithms such as artificial neural networks and support vector models. However, literature seems void of studies that test how the input data affects the accuracy of the machine-learning algorithm. It is possible to measure the far-field radiation pattern of earth-based antenna arrays, but very few researchers have validated their proposed techniques on a manufactured array. We therefore investigated the effects of various far-field sampling methods on the accuracy and training time of a feedforward neural network, and on the accuracies of different out-of-the-box classification algorithms, and the effect of the antenna array configuration on the accuracy of a support vector model. We simulated, manufactured and measured a 16-element circular patch antenna array to determine the feasibility of using the *simulated* far-field pattern as training data for a machine-learning algorithm designed to identify failures in a *measured* far-field pattern. We found it would not currently be feasible to employ machine learning to detect single element failures by measuring distortions in the far-field radiation patterns generated by a very large array of antennas in an irregular sparse configuration, such as those planned for the Square Kilometer Array (SKA) radio astronomy project.

UNIVERSITEIT VAN STELLENBOSCH

Opsomming

Fakulteit van Ingenieurswese
Departement van Elektriese en Elektroniese Ingenieurswese

Meesters in Ingenieurswese

Masjienleer vir Fout Analise op Antenna Samestellings

deur Lydia de Lange

Die gebruik van masjienleer word ondersoek vir tydige opsporing en uitsluiting van foutiewe elemente antenna samestellings. Die doel is om ’n betroubare manier te vind om foutiewe elemente vroegtydig op te spoor en uit te sluit, sodat die gemete data meer betroubaar sal kan wees, soos byvoorbeeld in groot antenna samestellings, soos die SKA radio astronomie projek. Vorige studies het bevind dat dit teoreties moontlik is om foutiewe elemente vanaf die ver-veld patroon te identifiseer deur die gebruik van ’n masjienleer algoritme soos ’n neurale netwerk (NN) of ondersteunings-vektor masjien ("SVM"). Daar is ’n tekort aan studies in die literatuur wat die invloed van die leerdata op die akkuraatheid van die masjienleer algoritme toets. Die ver-veld van ’n aardvaste antenna samestelling kan gemeet word, maar min navorsers het al hul voorgestelde metodes op ’n vervaardigde antenna samestelling getoets. In hierdie studie is daar ondersoek ingestel na die invloed van verskeie ver-veld steekproefmetodes op die akkuraatheid en opleidingstyd van ‘n FNN; en op die akkuraathede van verskeie standaard klassifiseringsalgoritmes; asook die invloed van die uitleg van die saamgestelde antenna op die akkuraatheid van ‘n SVM. ‘n 16-element sirkelvormige plak antenna samestelling is gesimuleer, vervaardig en gemeet, om vas te stel of die *gesimuleerde* ver-veldpatroon suksesvol gebruik kan word as leerdata vir ‘n masjienleer-algoritme vir foutsporing op ‘n *gemete* ver-veldpatroon. Ons sou nie tans aanbeveel om ’n enkele foutiewe element te probeer identifiseer deur versteurings te meet in die ver-veldpatroon van ‘n baie groot saamgestelde antenna met ‘n yl verspreide, ongeordende uitleg, soos beplan vir die SKA radio astronomie projek nie.

Contents

List of Figures

List of Tables

List of Abbreviations

ASKAP	Australian Square Kilometre Array Pathfinder
ASTRON	Netherlands Institute for Radio Astronomy
AVN	African Very Long Baseline Interferometry Network
CBF	Correlator/Beamformer
CEM	Computational Electromagnetic
DSN	Deep Space Network
EMBRACE	Electronic MultiBeam Radio Astronomy ConcEpt
FNN	Feedforward Neural Network
HartRAO	Hartebeesthoek Radio Astronomy Observatory
KAPB	Karoo Array Processor Building
LFAA	Low Frequency Aperture Array
LOFAR	LOw Frequency ARray
MeerKAT	Meer- Karoo Array Telescope
MFAA	Mid-Frequency Aperture Array
MID	Measurement IDentification Number
MLP	MultiLayer Perceptron
NN	Neural Network
NRF	National Research Foundation
PCB	Printed Circuit Board
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RFI	Radio Frequency Interference
SAAO	South African Astronomical Observatory
SALT	Southern African large telescope
SARAO	South African Radio Astronomy Observatory
SID	Scenario Identification Number
SKA	Square Kilometer Array
SMA	SubMiniature version A

SNR	Signal-to-Noise-Ratio
SVM	Support Vector Model
UHF	Ultra High Frequency
V-FASTR	VLBA Fast Radio Transients Experiment
VLBA	Very Long Baseline Array
VNA	Vector Network Analyzer

The financial assistance of the South African SKA Project (SKA SA) towards this research is hereby acknowledged (www.ska.ac.za)

1 Introduction

The title of this study is Machine Learning for Antenna Array Failure Analysis.

An antenna is defined in the IEEE Standard for Definitions of Terms for Antennas [1] as "That part of a transmitting or receiving system that is designed to radiate or to receive electromagnetic waves". An antenna array is defined as "An antenna comprised of a number of radiating elements the inputs (or outputs) of which are combined".

An antenna array could also be described as a group of antennas that observe electromagnetic waves in a certain direction at a certain centre frequency and bandwidth. Using interferometry, data from each antenna is collected, correlated and processed to produce an image of what the antennas observed at the given frequency. This image can be used by astronomers to observe astronomical objects.

Array failure. The failure of elements in an antenna array leads to distorted results. Currently, it is difficult to pinpoint a failed element in an array, especially in analogue beamforming antenna arrays, such as EMBRACE [2].

Array failure analysis. Antenna failures may go undetected until the severity of distortions makes it clear that an element (or more than one) has failed, and then astronomers and maintenance staff rely on manual inspection and other current methods [3]–[6] to identify any element that needs to be replaced.

An improved ability to remotely detect failed elements and locate their exact positions could improve data reliability for astronomers, and enable more efficient approaches to repairs and maintenance. This would be particularly beneficial for large aperture arrays such as the SKA [7] radio astronomy project and similar organisations and situations.

The certainty that an antenna array is fully functional would strengthen confidence in the validity of recorded results; and secondly, once a failed element is identified, methods for correcting the antenna failure [8]–[11] could be applied to exclude any identified element from calculations. Thus, observed data could be salvaged.

Machine-learning methods such as SVMs [12] and neural networks [13] hold promise as alternative methodologies for this purpose. Previous work has shown that failed elements can be detected

FIGURE 1.1: The LOFAR antenna array, a pathfinder for the SKA, operates at the lowest frequencies observable from earth. LOFAR consists of 50 Stations across Europe, with the core (38 stations) concentrated in the Netherlands [17].

by analysing the far-field radiation patterns generated by an antenna array, as will be discussed in Section 1.2.

This study investigated whether a machine-learning method could be used to identify failed elements in a large aperture array, such as LOFAR [14], [15] (shown in Fig. 1.1) or the LFAA and MFAA [16] planned for the SKA radio astronomy project. The method proposed in this thesis is focused on analogue aperture arrays. A dish antenna is a complex structure and is very likely to have monitoring systems in place to detect failures in the system. The method proposed in this thesis is more appropriate for analogue aperture arrays, where each individual element is not monitored. The investigation was timed to precede the design and construction of the SKA radio astronomy project, so that it can be considered in system health management conceptualisation.

1.1 Background

1.1.1 The Square Kilometer Array

The SKA is an international project to build the world's largest radio astronomy interferometer yet, with a total collection area of one square kilometre.

Building a radio telescope with a total collecting area of one square kilometre requires stations across the globe. Since the idea was conceived in 1991, 12 member countries (Australia, Canada, China, France, India, Italy, New Zealand, South Africa, Spain, Sweden, the Netherlands, and the United Kingdom), about 100 organisations and 20 partner countries have joined forces to bring the project to fruition. The huge collecting area and long distances between stations will allow the SKA to produce high-resolution images of large parts of the sky at once, with a higher sensitivity and angular resolution, at a faster rate, than ever before.

The SKA is planned to have a frequency range from 50 MHz to 14 GHz. The range will be in two parts: SKA-low and SKA-mid. SKA-low will consist of almost a million small, low-frequency antennas, operating from 50 MHz to 350 MHz. This array is referred to as the LFAA. SKA-mid will operate from 350 MHz to 14 GHz, and will consist of thousands of antennas, including the MeerKAT dishes and the MFAA [16].

The SKA will be built in two phases - SKA1 and SKA2. During phase one, about 10% of the total collecting area will be built in South Africa and Australia. These locations are in some of the most sparsely populated areas on earth, with some of the lowest levels of man-made RFI. The locations were also chosen for the drier atmospheric state above the sites and because their position in the southern hemisphere provides a good view of the Milky Way galaxy. SKA1-low will be built in the Murchison Shire of Australia. SKA1-mid will be built in the Karoo of South Africa. SKA2 will be the completion of the full array, expanding SKA1-mid into African partner countries, and SKA1-low into other parts of Australia [16].

The technology, systems and techniques required to build the SKA are being developed and refined around the world by partner organisations. Their projects fall into three categories: design studies, pathfinders and precursors. System end technology prototypes are designed and tested in design studies. Precursors (e.g., ASKAP [18], MeerKAT [19]) and pathfinders (e.g., LOFAR [14], [15], EMBRACE [2]) are demonstrator telescopes and systems built on the SKA sites and in partner countries respectively. The experience gained from building and maintaining these telescopes will be crucial when finally designing, building and operating the SKA.

Once the SKA telescope comes online, unprecedented scientific pursuits in the fields of cosmology, fundamental physics, astrophysics and particle astrophysics will become possible. Scientists will use the SKA to test Einstein's theory of relativity and try to answer questions about cosmology, the formation and evolution of galaxies, and the nature of dark matter and dark energy. The SKA will make it possible to investigate cosmic magnetism and the epoch of re-ionisation. The SKA plans to expand the observable universe and extend the search for extra-terrestrial life [16].

The detection of failed elements in this large antenna array is important to keep the interferometer as sensitive as possible. It was therefore considered worth investigating machine learning as a tool to

identify failed elements in the SKA.

Machine learning has been called the fourth industrial revolution and other researchers have used it to identify failed elements in an antenna array, as will be discussed in Section 1.2. However, most previous work focused on identifying failed elements from the far-field pattern of the antenna array (i.e., the geometric representation of the electric field strength at a constant distance far away). There was not enough focus on using machine learning to detect element failure in radio astronomy specifically.

For this reason, the current project has been studying the use of machine-learning methods to detect element failure in that area for the 24 months since the end of 2017. At the same time, the Netherlands Institute for Radio Astronomy (ASTRON) [20] has been working on a related project, using an unsupervised learning method to cluster data from LOFAR.

1.1.2 The SKA in South Africa and at Stellenbosch University

The SKA in South Africa is led by SARAO [21], one of seven National Research Facilities of the National Research Foundation (NRF) of the Department of Science and Technology [22]. These seven facilities are organised into four national infrastructure platforms: dealing with biodiversity, nuclear matters, databases and astronomy [23]. The astronomy platform consists of SARAO and the SAAO. The SAAO is responsible for optical and infrared astronomy, focusing on astronomy and astrophysics, including oversight of the Southern African large telescope (SALT) [24] at the SAAO site in Sutherland. SALT is the largest single optical telescope in the southern hemisphere, and among the largest in the world.

The SARAO spearheads South Africa's engineering, science and construction activities in the SKA, and incorporates radio astronomy instruments and programmes, such as the MeerKAT and KAT-7 telescopes in the Karoo, the HartRAO in Gauteng, and the AVN in nine African countries [21].

The MeerKAT telescope, inaugurated in July 2018, is the SKA precursor at the SKA-mid site in the Karoo, which will eventually constitute 25% of SKA1-mid. Of MeerKAT's existing 64 antenna dishes, 61% are within 500 m of the centre, while the remaining 39% extend to 4 km from the centre. The addition of 133 antennas to the MeerKAT precursor telescope's 64 dishes will form an array of nearly 200 dishes, where the longest baseline will be 150 km [25]. An illustration of the SKA1-mid antenna positions is shown in Fig. 1.2. This large footprint of the SKA1-mid in South Africa poses major challenges for maintenance operations. The positions of the SKA1-mid antennas are shown in Fig. 1.2. SKA1-mid is just a part of the larger SKA radio astronomy project, which demonstrates the size of the project.

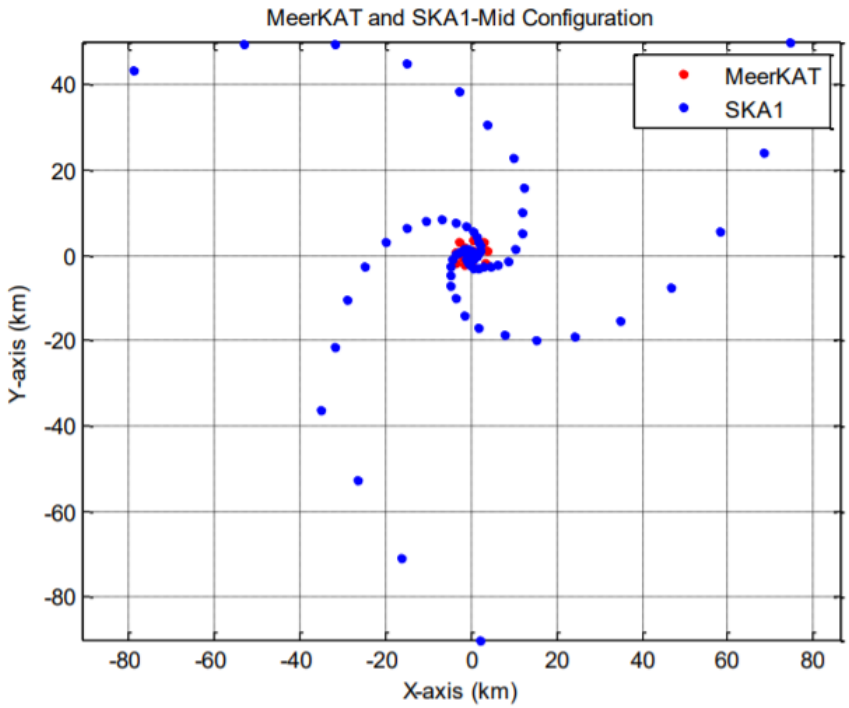


FIGURE 1.2: Illustration of the positions of the SKA1-mid antennas, relative to the first antenna in the array. The existing 64 MeerKAT antennas are shown in red, and the 133 additional SKA1-mid antennas in blue. [25].

The data observed by each antenna is digitised immediately at the antenna. The digital data streams are sent to the on-site Karoo Array Processor Building (KAPB) and processed by the CBF digital signal processor. The processed data is stored at the KAPB.

The SARAO control centre and the Centre for High Performance Computing (CHPC) are in Cape Town, approximately 600 km away. A long-haul optical fibre provides a connection between the KAPB and these centres, to transfer data and control and monitor operations.

Herein lies the reason for the investigations reported in this thesis: The Karoo is a very hot, windy, desolate place, where antenna failures are to be expected, and the distance from SKA1-mid to the SARAO control centre makes maintenance a challenge.

Stellenbosch University's Electrical and Electronic Engineering Department hosts the Chair in Antenna Systems for the SKA, one of the five research chairs for the SKA SA have currently been allocated to South African universities [26]. Projects under this chair perform research into various fields, focused specifically for radio telescope applications. The projects are not limited to the design of physical components, such as the design of reflector feeds or antenna arrays. They also encompass research and design of software, such as the optimisation of the modelling routines that were used to design antennas.

SARAO's science data processing team is already procuring machine-learning expertise for a range of applications, including RFI characteristics and detection, anomaly detection, and radio source identification and classification [27].

This provides the context for a research project by Dr DJ Ludick, under the SKA SA Chair at Stellenbosch University. This thesis is the first of a series of theses that will investigate the use of machine learning for array failure analysis in the context of the SKA. This work focused on a part of Dr DJ Ludick's larger project, and investigated machine-learning algorithms and the use of the far-field pattern as training data for the machine-learning algorithm.

The training data required to train machine-learning algorithms was taken from a large distributed array (for example, the beam pattern of the array) for all relevant element failure scenarios. This training data was simulated with the computational electromagnetic software, FEKO.

1.2 Literature Synopsis

This is an interdisciplinary study, that requires insights from the fields of machine learning and antenna theory. This section provides a quick overview of how the processes introduced in this text has been applied in both technical fields. A more focused overview will be presented about antenna theory and machine learning concepts, respectively. Antenna theory concepts, such as the far-field radiation pattern, are explained in more detail in Chapter 2. Machine-learning concepts such as training, SVMs and FNNs are discussed in Chapter 3.

Software [28] and hardware [3]–[6], [29] techniques have been used in the past to detect failed elements in an antenna array. The idea of using machine learning to identify failed elements in an antenna array is not novel. Element failure in antenna arrays is a problem that researchers have addressed with machine-learning techniques in military [30], satellite [31], [32] and other wireless communication applications [33].

In the next two sections, we describe how machine learning has been used previously in radio astronomy as a whole, and more specifically to detect element failure in a radio antenna array. We also look at how machine-learning experts in other fields have detected failed elements in an antenna array.

1.2.1 Machine learning in radio astronomy

In radio astronomy, machine learning is mainly being used for real-time analysis of large data volumes, such as will be produced by the SKA [34], and to passively analyse data streams to identify anomalous events that might be of interest to scientists. An example of the latter [35] is the V-FASTR experiment at the VLBA, which uses machine learning to search for anomalous fast radio transients and has been

running since July 2011. Another use for machine learning in radio astronomy is in the detection and classification of RFI in the observed data [36].

An example of using machine learning for system diagnostics in the context of radio antenna systems was a discussion of a complex set of diagnostic tools developed by NASA for monitoring, analysing and diagnosing their ground systems and spacecraft [37].

NASA's Jet Propulsion Laboratory developed two complex software tools to reduce operational and maintenance costs, BEAM and SHINE. The design of the tools, originally developed to maintain the system health of spacecraft, allowed them to be applied to complex new systems. These tools were repurposed for fault diagnostics and prognosis of the DSN [37], which is an international network of antennas. The DSN tracks spacecraft and performs radio and radar astronomy observations.

Like the SKA, the DSN is required to have minimum downtime and system failures. The article did not go into the detail of data collection methods, or data types. It only stated that BEAM used raw sensor data and software-derived data currently used by human experts to perform diagnostics, along with all DSN data necessary to perform fault diagnostics. The tools provided new insights into system visibility that were not possible with the previous channel-based diagnostic techniques.

BEAM used algorithms such as stochastic modelling and non-linear information filtering, and processed data using temporal channel analysis and adaptive wavelet theory. SHINE used the outputs of BEAM, along with model-based reasoning and case-based reasoning, to monitor the DSN.

1.2.2 Machine learning to detect antenna array element failure in other fields

Antenna arrays, whether used for military applications, in space or on aircraft, are all prone to element failure. Machine-learning researchers have been tasked with identifying these failed elements.

As will be shown in Chapter 4, antenna array element failure leads to distorted far-field radiation patterns. In many of the articles reviewed, the defective elements were found so that the remaining functional elements' excitations could be adjusted in software to restore the radiation pattern as closely as possible to its original form. This was done by so-called array failure correction [8]–[11], which is primarily applicable to active antennas. It is especially useful for antenna arrays that have been deployed in space and are impossible to reach and replace.

In radio astronomy, some interferometers are capable of doing active beamforming through software to compensate for failed elements in the antenna array. This is useful if an antenna in the array cannot be replaced before an observation has to be made. However, as for aircraft antennas, the main purpose for element failure detection in radio astronomy is to enable a failed element to be replaced as soon as possible. The instrument is used for observations that should be as sensitive as possible, and can only function properly with a full complement of elements.

The most common technique for training machine-learning algorithms described in the literature currently available, is to train them on the amplitude of the far-field pattern of an antenna array [32], [33], [38]–[41]. As far back as 1997, Rochblatt [42] described a method of measuring the far-field of reflector antennas fixed on earth, so it is feasible to use the entire far-field pattern as training data. However, measuring the complex far-field pattern requires expensive equipment and a reference signal, so many researchers used only the amplitude of the far-field pattern [33]. Far less frequently, machine learning was applied to the near-field data of the antenna array [43]. It has also been suggested [40] that the scattering parameters of the antennas in the antenna array could be used as training data.

Most researchers used a linear array as their training data [8], [38], [40], [41], and recommended expanding the work to planar arrays, as was done by Bucci [33].

When using the far-field data as training data, most articles reported using a cut of the far-field pattern in the ϕ angle [30], [39], [40]. Some authors kept the simulation of the element failures simple, so that antennas were either completely ON or OFF [30], [33], [39], while others also investigated partial failures [38], [40], [41]. Their far-field radiation pattern was either simulated with commercial software to include the effects of mutual coupling, or was calculated with self-written software, after which authors recommended that mutual coupling should be taken into consideration in future work. Researchers rarely simulated all failure scenarios. This was only done when the array was very small, with four or five elements in the antenna array [38]. With larger arrays, the convention has been to simulate up to three or four simultaneous failures. Several authors applied some post-processing to the generated far-field patterns, such as adding noise to the signal, to increase the number of training examples given to the algorithm to train on [30].

The algorithms reportedly were genetic algorithms [31], [33], [39], [43], artificial neural networks [38], [44], case-based reasoning [31], [32], support vector models [30], [40], the Woodward-Lawson method [32], and bacteria foraging optimisation [41].

Very few researchers have validated their proposed techniques on a manufactured array. However, Miao et al [39] used a genetic algorithm to detect four unique failure scenarios in a linear 32-element microstrip printed dipole array. The improved success rate was attributed to the fact that mutual coupling was taken into account during the simulation of the far-field patterns.

1.3 Objectives

The objectives of this work are set out below.

- Use an FNN to rank different ways of sampling the far-field radiation pattern by how they increase the accuracy of the machine-learning algorithm.

- Evaluate which out-of-the-box scikit-learn [45] algorithms perform best on which training data sets, to validate the results that were obtained on the FNN and to see which training algorithms work better than the FNN.
- Use the technique described in [30] to investigate the effect of the array configuration on the accuracy of an SVM.
- Test the effect of the signal-to-noise-ratio (SNR) on the accuracy of an SVM.
- Manufacture and measure an antenna array to see whether it is practical to use its far-field radiation pattern to identify failed elements, and evaluate whether the simulated far-field pattern resembles the measured far-field pattern closely enough to provide training data for an algorithm that must classify the failed elements in a measured far-field pattern.

1.4 Contributions

Two articles were written for international peer reviewed IEEE conferences:

- L. de Lange and D. J. Ludick, "Application of Machine Learning for Antenna Array Failure Analysis," *2018 International Workshop on Computing, Electromagnetics, and Machine Intelligence (CEMi)*, Stellenbosch, 2018, pp. 5-6 [46]; and
- L. de Lange, D. J. Ludick and T. L. Grobler, "Detecting Failed Elements in an Arbitrary Antenna Array using Machine Learning," *2019 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, Granada, Spain, 2019, pp. 1099-1103 [47].

The following contributions to the current body of knowledge emanated from this work, by completing the objectives listed in Section 1.3.

- We investigated different ways of sampling the far-field radiation pattern and found it is best to sample the far-field pattern in a 3-D grid, rather than sampling the far-field pattern only in the $\phi = 0°$ cut, with few samples, as was common practice in previous work. A data set of training data sampled in a 3-D grid pattern contains more information to discriminate between failure scenarios. This means that fewer samples have to be taken of the far-field radiation pattern to form a reliable training data set.
- We investigated different array configurations, as was recommended by previous work, and found that the configuration of the array does indeed have an effect on the accuracy of the

trained model. When an array is symmetrical in a plane, the far-field radiation pattern should not be sampled in that plane. In symmetrical configurations such as a linear or regular dense array, taking a single cut through the origin (such as the $\phi = 0°$ cut or the $\phi = 90°$ cut), will result in some classes having identical patterns wherever the OFF antennas are a mirror image of each other. This makes it impossible for any classification algorithm to achieve high accuracy.

- We found that adding noise to a pattern with various signal-to-noise ratios does not model the difference between measured and simulated data sufficiently. Therefore noisy signals should not be used to generate more training data for the machine-learning algorithm if the purpose is to identify the failed elements in a measured signal. Future work should rather simulate an array configuration many times per failure scenario. Each time the scenario is simulated, small random deviations can be made in parameters that have an effect on the shape of the pattern (e.g., antenna geometry and material properties).

- We confirmed the rule that the training data for a machine-learning algorithm has to resemble the testing data for a machine-learning algorithm to be able to classify the testing data. If the far-field radiation pattern is chosen as the data that will be given to the machine-learning algorithm, the testing data to classify failure scenarios for an SKA aperture array. The training data must then resemble the measured LFAA far-field radiation pattern to be able to classify an unseen case.

- The core finding of this study is that it would not be feasible, at this point in the evolving history of technology, to employ machine learning to detect single antenna failures by measuring distortions in the far-field radiation patterns generated by a very large array of antennas in an irregular sparse configuration, as planned for the SKA radio astronomy project.

1.5 Overview of This Work

Chapter 1 explained the background to this study of machine learning for antenna array failure analysis, defined terms, and summarised the literature about it. It clarified the objectives of the study and how it has already contributed to the body of knowledge on the subject.

Chapter 2 explains the antenna theory necessary to understand the rest of the thesis. The most important concepts are the far-field radiation pattern, the reflection coefficient (commonly referred to as the S11 parameter), and antenna arrays. The chapter discusses the history of antennas, the principles of antenna radiation, and how and why antennas are arrayed.

Chapter 3 explains how data was gathered from antenna arrays simulated in FEKO [48], and used to generate data sets that were used for machine-learning experiments. The chapter discusses the two machine-learning methods used in this work - FNNs and SVMs.

Chapter 4 describes how we investigated the effect of the choice of training data on a classification algorithm's accuracy. Two experiments were done, using the training data and the machine-learning methods described in Chapter 3. The two experiments are summarised in Table 4.1. In the first experiment, the far-field radiation pattern of a 5×5 bow-tie antenna array was sampled in different ways. The samples were processed as described in Chapter 3, and used to train an FNN and other classification algorithms. In the second experiment, the methods introduced by Yeo [30] were investigated for different antenna array configurations. First, the SVM kernel that achieved the highest accuracy at the lowest SNR was selected. The results of the selected SVM on different array configurations were compared to determine the effect of the number of elements, number of classes and layout of an array configuration on the accuracy of the SVM.

Chapter 5 relates how a 16-element circular patch antenna array was designed, manufactured and measured. The simulated and measured far-field patterns of the antenna array were compared to determine whether it is feasible to use the *simulated* far-field pattern as training data for a machine-learning algorithm intended to identify failures in a *measured* far-field pattern.

Finally, general conclusions are drawn in Chapter 6, and some recommendations for future work are made, based on the contributions summarised in Section 1.4.

2 Overview of Antenna Concepts

An antenna is a structure that transfers radio waves or electromagnetic energy between free space and electronics. It can either transmit, where radio waves are generated, or receive, where radio waves are observed.

This chapter addresses antenna theory, based on work by Balanis [49], [50]. We discuss the interesting history of antennas; we explain antenna radiation based on an 8-element linear bow-tie antenna array; we clarify how antenna arrays work and why they are useful in applications such as radio astronomy; and, finally, we describe important antenna parameters and figures of merit. There is special focus on concepts important to this thesis, such as antenna arrays and the far-field radiation pattern that will be used later in Chapter 3.

2.1 History of Antennas

In 1873, James Clerk Maxwell published his work [51] in which he related the theories by Faraday and Ampere on magnetism and electricity, through a set of four equations known as Maxwell's equations. He also proved that light was an electromagnetic phenomenon. The famous four equations are shown below in integral form [50].

Gauss's law for electric fields states that the integral of the electric field ($\vec{E}$) exiting an area enclosing a volume ($\vec{A}$) is equal to the total charge (q) inside the volume.

$$\int \vec{E} \cdot d\vec{A} = \frac{q}{\epsilon_0}. \tag{2.1}$$

Gauss's law for magnetic fields states that the integral of the magnetic field ($\vec{B}$) exiting an area enclosing a volume ($\vec{A}$) is equal to zero.

$$\int \vec{B} \cdot d\vec{A} = 0. \tag{2.2}$$

Faraday's law of magnetic induction states that the integral of the electric field ($\vec{E}$) around a closed loop is equal to the total change in voltage. This change in voltage is brought about by a varying

magnetic field ($\vec{B}$) passing through the circuit.

$$\oint \vec{E} \cdot d\vec{l} = -\frac{d}{dt}\left(\int \vec{B} \cdot d\vec{A}\right). \tag{2.3}$$

Ampere's law with Maxwell's displacement current gives the total magnetic force around a circuit as a sum of the current (I) through the circuit and the displacement current, $\frac{d}{dt}\left(\epsilon_0 \int \vec{E} \cdot d\vec{A}\right)$, which is any varying electric field through the circuit.

$$\oint \vec{B} \cdot d\vec{l} = \left(I + \frac{d}{dt}\left(\epsilon_0 \int \vec{E} \cdot d\vec{A}\right)\right). \tag{2.4}$$

These equations can be cast into a vector wave equation, the solutions to which are propogating electromagnetic waves. Heinrich Rudolph Hertz was the first to demonstrate that wireless communication was possible using these electromagnetic waves in 1886. In 1901, Guglielmo Marconi was the first to demonstrate transatlantic transmission with wire antennas. In 1933, Karl Guthe Jansky discovered that the Milky Way galaxy emits radio waves, which is why he is known as the father of radio astronomy. Jansky used the rotating antenna shown in Fig. 2.1, nicknamed "Jansky's Merry-go-round" to survey the sky at a frequency of 20.5 MHz [52].

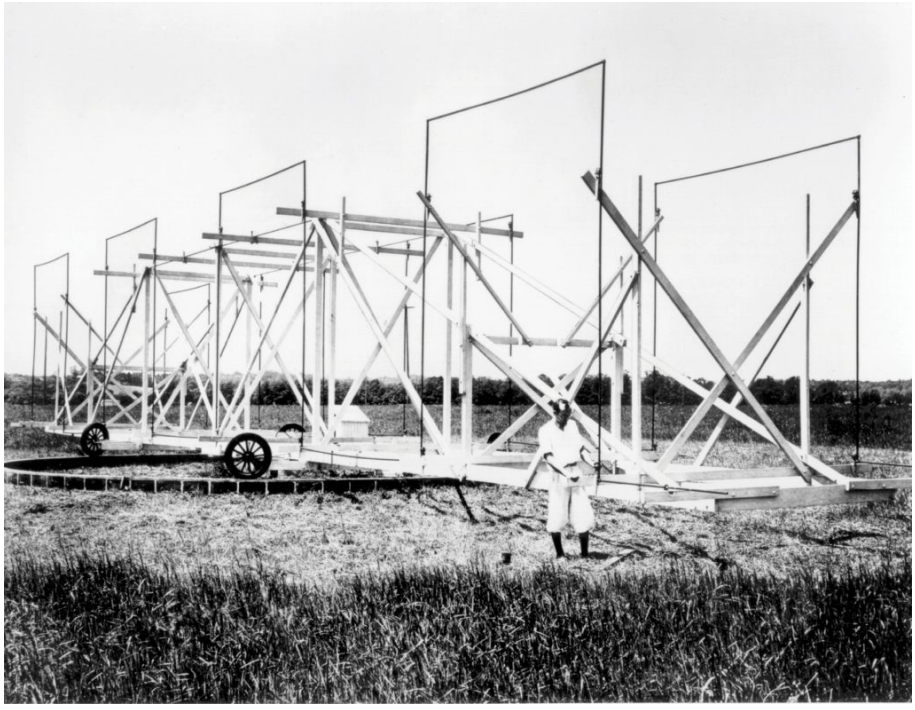


FIGURE 2.1: The rotating antenna Jansky built, nicknamed "Jansky's Merry-go-round". The antenna functioned at 20.5 MHz. [52]

It was during World War II that antenna design became more prominent, and the focus shifted from designing antenna systems to improving the structure and material of the antennas [53]. Until then, the radiating element in a wireless communication system was a wire antenna, and communication was limited to up to the ultra high frequency (UHF) band, spanning 300 MHz to 3 GHz. During the war, modern radiating elements were introduced, such as horn antennas, reflector antennas, and waveguides [54]. Nevertheless, the design of the antenna was not the largest concern in the overall system design. Antennas were designed in a "cut and try" fashion [49]. However, as computers became more advanced, it was finally possible to invent numerical methods and simulation software [48], [55] to design antennas, so that the antenna is now the most significant part in the system design.

Today, antenna designs can be very complex and large. In contrast to the first radio antenna for radio astronomy, the Five-hundred-meter Aperture Spherical Telescope (FAST) [56], is shown in Fig. 2.2 to demonstrate how much has been achieved in the field of radio astronomy in less than 100 years. FAST was installed in 2016 in China and is the world's largest single-aperture telescope.

FIGURE 2.2: The world's largest single-aperture telescope, the Five-hundred-meter Aperture Spherical Telescope (FAST), was installed in 2016 in China [57].

2.2 Antenna Radiation

Antennas are designed for specific tasks. These tasks usually require the antenna to be directional, which means that the radiation pattern is optimised to be sensitive in a certain direction and suppressed in other directions. In other words, design specifications usually require that an antenna receives a signal from a certain direction and is not sensitive to signals from other directions. To achieve this, the radiation pattern is optimised by using the correct antenna type and shape.

The antenna radiation pattern is a graphical representation of an antenna parameter and is usually shown in the far-field region. Parameters that can be expressed in a radiation pattern include the amplitude and phase of the electric or magnetic field. When referred to in this work, the antenna's radiation pattern is a geometric representation of the amplitude of the electric field strength at a constant distance far away from the antenna. The representation is usually plotted in a spherical coordinate system, as a function of the orthogonal (perpendicular) directions θ and ϕ. In this work, the radiation pattern is always normalised and plotted on the logarithmic scale. The orthogonal directions are demonstrated in Fig. 2.3.

2.2.1 Electromagnetic field regions for an antenna

The characteristics of the electromagnetic fields generated by an antenna change as the distance from the antenna increases. They are therefore divided into three regions – the reactive near-field, the radiating near-field, and the far-field region. The electric and magnetic fields must be in phase and perpendicular (orthogonal) in order to propagate or radiate [50].

The near-field of an antenna is divided into reactive and radiating regions. The reactive region is the closest to the antenna. Here, the electric and magnetic fields are 90 degrees out of phase, which means that the antenna is not radiating. The region is generally defined by the equation

$$d < 0.62\sqrt{\frac{D^3}{\lambda}}, \tag{2.5}$$

where d is the distance from the antenna, D is the diameter of the smallest sphere containing the antenna, and λ is the wavelength calculated at the operating frequency, $\lambda = \frac{c}{f}$, with c as the speed of light ($2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$) and f in Hz.

In the radiative near-field (or Fresnel) region, the electromagnetic fields start changing from reactive fields to radiating fields. In this region, the shape of the radiation pattern can be ever-changing as the distance from the antenna increases. The radiative near-field exists in the region defined by the boundary

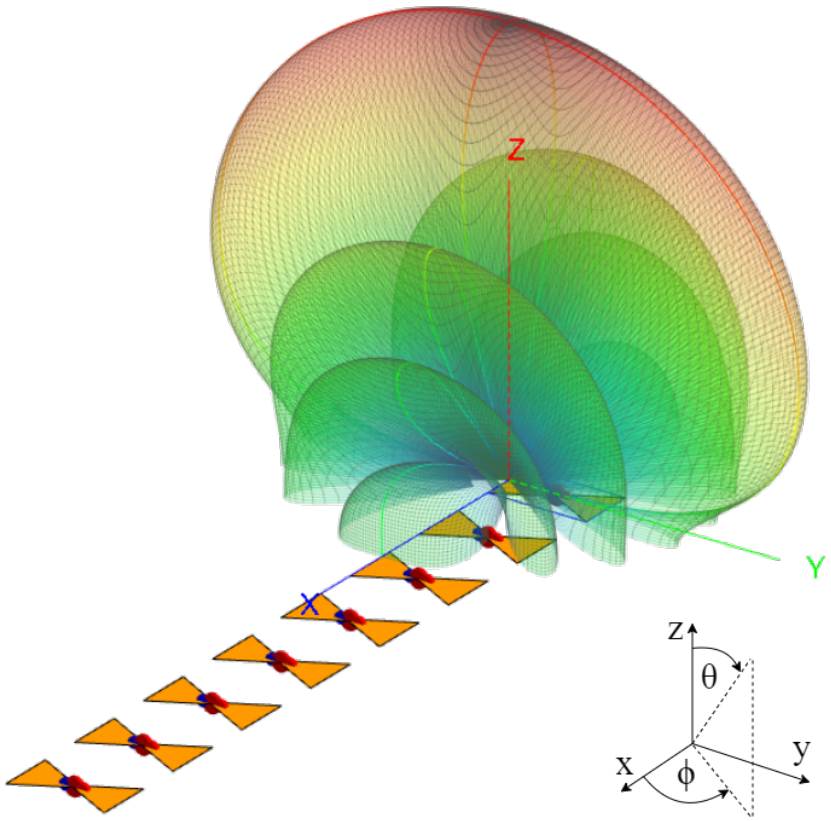


FIGURE 2.3: An 8-element linear bow-tie antenna array and its far-field radiation pattern. The orthogonal directions θ and ϕ are demonstrated.

$$0.62\sqrt{\frac{D^3}{\lambda}} < d < \frac{2D^2}{\lambda}. \tag{2.6}$$

The far-field (or Fraunhofer) region is where long-distance antennas, such as those used for radio astronomy, operate. In this region, the electric and magnetic fields are orthogonal to both each other and the direction of propagation, and therefore radiating. The antenna is far enough away from the observation point that the fields can be observed as plane waves. In the far-field region, as the distance from the antenna increases, the fields and power density decrease, but the geometric shape of the radiation pattern remains the same. The far-field exists in the region

$$d > \frac{2D^2}{\lambda}. \tag{2.7}$$

As the size and operating frequency of an antenna increases, the far-field region quickly becomes

too far away to measure at a measurement facility such as the anechoic chamber at Stellenbosch University. Chapter 19 of [58] explains a mathematical transformation to determine the far-field from a near-field measurement.

2.2.2 Antenna arrays

Some applications have requirements for radiation characteristics that cannot be achieved by a single antenna, but may be achievable when multiple antennas are arranged in a certain configuration. The SKA is an example of such an application.

An antenna array is a group of antennas placed in a specific configuration. The antennas are designed for a specific frequency range. Antennas are placed in an array to improve the performance of that of a single antenna. Antennna arrays can be used to increase the overall gain, create a narrower main beam, cancel out interference from specific directions, find the direction of arrival of an observed signal or steer the beam electronically (without moving any mechanical parts). Interferometry is used to collect, correlate and process the observations by all the individual antennas. In this way, a higher resolution image is produced, which is very useful for radio astronomy.

Arrays can be dense or sparse, and regularly or irregularly spaced. Three different antenna array configurations are investigated in Section 4.3.

Many antenna arrays have been designed as pathfinders for the SKA. One example is the analogue beamformed phased array, EMBRACE. The output signals of its elements, shown in Fig. 2.4, are combined in analogue and then the correlated signal is digitised and processed [2]. As mentioned in Chapter 1, it can be particularly difficult to pinpoint a failed element in an analogue beamformed array. Because the output of an individual antenna is not digitised and sent to the correlator directly, antenna elements are less likely to be monitored individually. A machine-learning technique that can identify element failures would therefore be very useful.

2.3 Antenna Parameters and Figures of Merit

Many parameters and figures of merit have been established to characterise the performance of antennas, but only the figures of merit important to this thesis are shown in this section. A more detailed discussion of antenna parameters and figures of merit can be found in [50].

The 8-element linear bow-tie antenna array, shown above in Fig. 2.3, is used in this section to demonstrate some concepts of radiation patterns, while Fig. 2.5 demonstrates some derived quantities and other attributes of radiation patterns. The pattern in Fig. 2.5 is the $\phi = 0°$ cut of the electric far-field radiation pattern shown in Fig. 2.3.

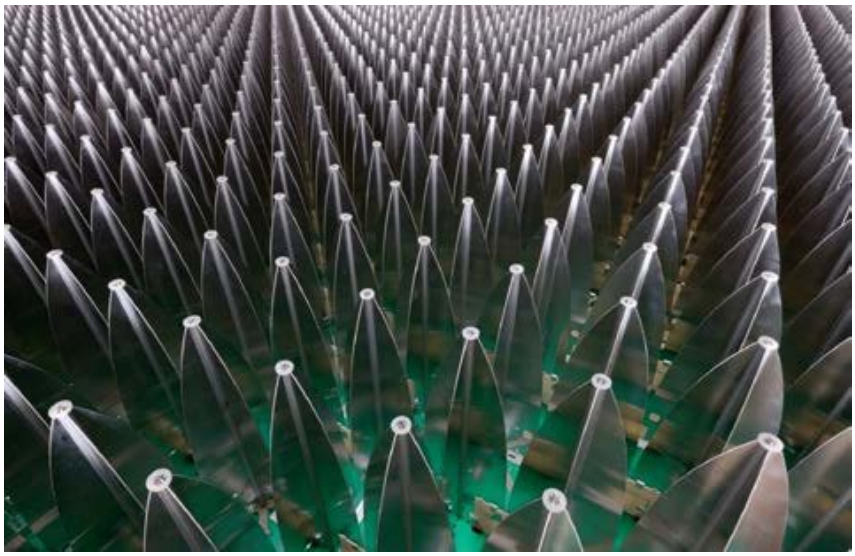


FIGURE 2.4: A photograph of the aluminium Vivaldi elements of one of the EMBRACE stations [2].

Lobes. The *main lobe* of the radiation pattern contains the direction of maximum radiation. The *side lobes* are radiation lobes in any other direction than that of the main lobe. The *side lobe level* is the amplitude of the side lobe relative to the amplitude of the main lobe. Any lobes besides the major lobe are also commonly referred to as *minor lobes*.

The radiation intensity [1] is "the power radiated from an antenna per unit solid angle (steredian)". The angle between two directions in which the radiation intensity is equal is defined as the *beamwidth* (BW). The *half-power beamwidth* (HPBW) is where the radiation intensity is half the maximum radiation intensity.

The reflection coefficient is defined at the terminals of an antenna and quantifies the ratio of the reflected to the transmitted power. The reflection coefficient, denoted by Γ, is defined as

$$\Gamma = 20 \log_{10} \left(\frac{V^-}{V^+} \right), \tag{2.8}$$

where V^- is the reflected voltage wave and V^+ is the transmitted voltage wave. In the context of an array environment the reflection coefficient is measured for a particular element, with other elements terminated in a load impedance. A plot of the magnitude of the reflection coefficient vs frequency can be used to define the bandwidth of the antenna. A threshold of $|\Gamma| \leq -10$ dB is typically used, which is illustrated in Fig. 2.6.

2.4 Conclusion

Important concepts relating to antenna theory in the context of radio astronomy were discussed in this chapter. In the next chapter, we show how the electrical far-field radiation pattern is processed into training data sets. The chapter also describes the machine-learning algorithms that were trained on far-field data sets.

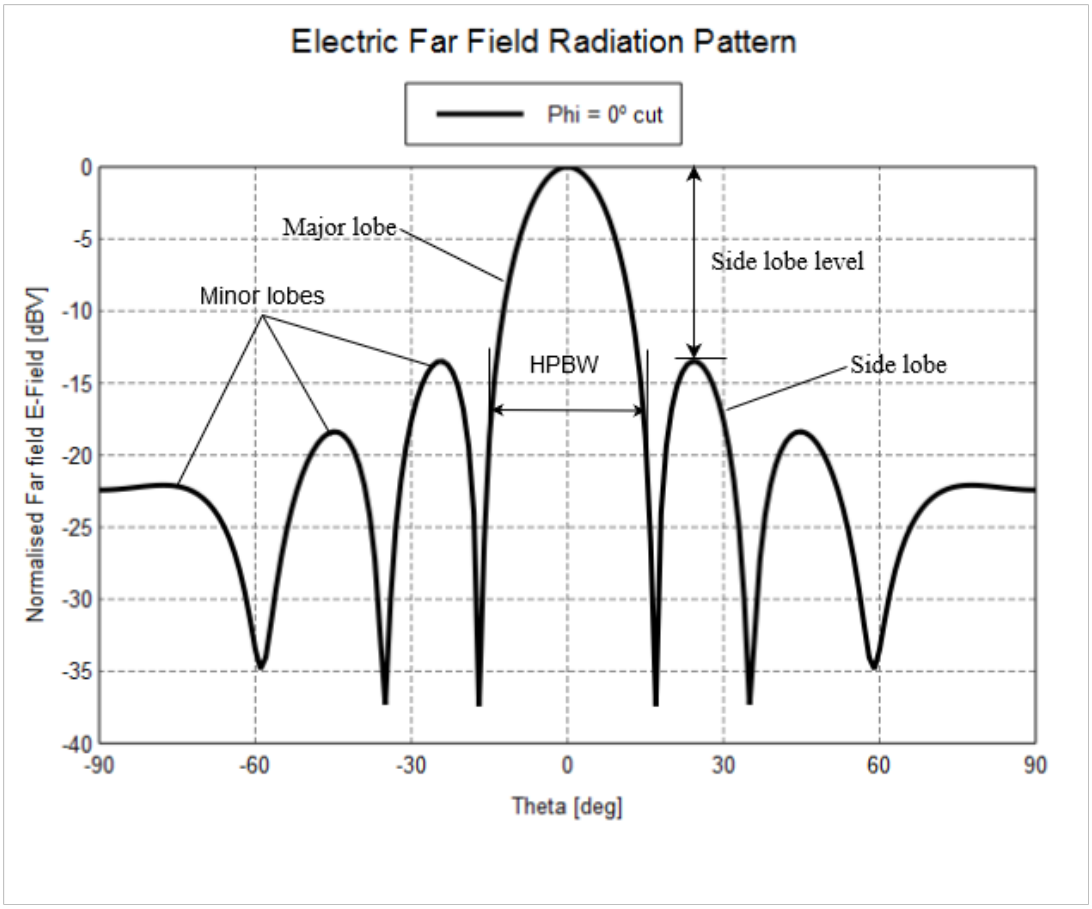


FIGURE 2.5: The $\phi = 0°$ cut of the electric far-field radiation pattern or the 8-element linear array shown in Fig. 2.3. Some figures of merit and derived quantities described in the text are indicated.

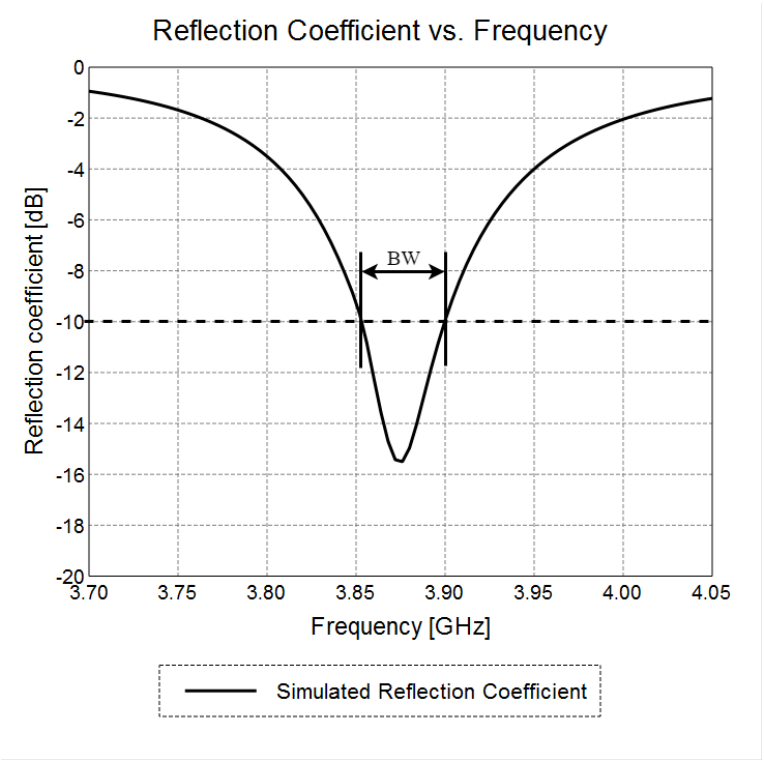


FIGURE 2.6: The magnitude of the reflection coefficient of an antenna vs frequency can be used to define the bandwidth (BW) of the antenna. At a threshold of $(\Gamma) = -10$ dB, the bandwidth above will be equal to 50 MHz.

3 Machine Learning for Failure Detection

The previous chapter introduced antenna arrays and far-fields. In this chapter, we describe how we applied machine learning to the far-field pattern of an antenna array, in order to locate failed elements in the array. The two main machine-learning algorithms used were FNNs and SVMs. The data sets used to train the machine-learning algorithms were the far-field patterns of various unique failure scenarios for a certain array configuration. We describe the different data set types, how machine-learning algorithms were trained, and how the far-field pattern was generated and processed into data sets that could be used by the machine-learning algorithm.

3.1 Data Set Generation

Consider the 8-element linear antenna array in Fig. 3.1.

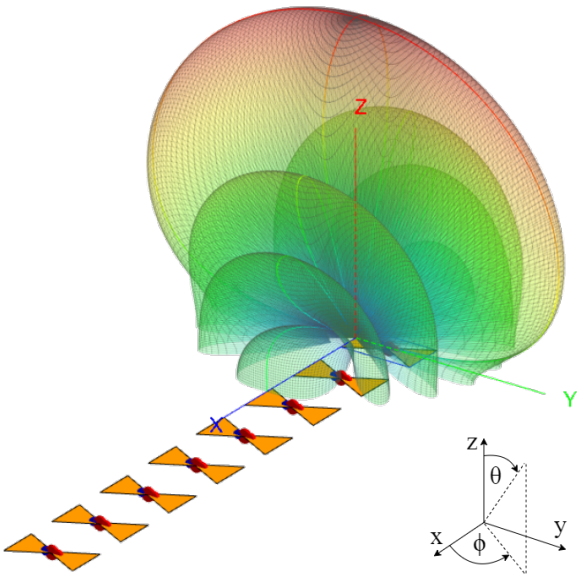


FIGURE 3.1: 8-element linear bow-tie antenna array, showing only the $\phi = 0$ cut.

This array and its far-field patterns are used throughout the section to exemplify the explanations. Given the number of elements, n, the number of unique failure scenarios, R, that can occur for this antenna array, can be calculated as

$$R = 2^n. \tag{3.1}$$

For the 8-element linear array, this corresponds to 256 unique failure scenarios. Each far-field pattern is used as an entry in the training data set. In Fig. 3.1, the whole far-field geometrical pattern is shown; and the $\phi = 0$ cut is highlighted.

Many data sets can be generated from one antenna array configuration. The data sets correspond to the different sampling methods used to sample the far-field pattern of the antenna array. The different data sets are summarised in Table 3.1, where the Description column describes how each failure scenario is sampled.

TABLE 3.1: Data sets formed from the 8-element bow-tie antenna array

Data set	**Description**
$\phi = 0°$	The $\phi = 0°$ cuts of each failure scenario
$\phi = 90°$	The $\phi = 90°$ cuts of each failure scenario
Principle cuts	The principle cuts of each failure scenario, a combination of $\phi = 0°$ and $\phi = 90°$
$\phi = 45°$	The $\phi = 45°$ cuts of each failure scenario
$\phi = 135°$	The $\phi = 135°$ cuts of each failure scenario
Diagonal cuts	The diagonal cuts of each failure scenario, a combination of $\phi = 45°$ and $\phi = 135°$
All cuts	All 4 cuts of each failure scenario, a combination of the principal and diagonal cuts
3-D, 180 samples	A 3-D sampled grid of the far-field pattern, with 180 samples
3-D, 360 samples	A 3-D sampled grid of the far-field pattern, with 360 samples
3-D, 720 samples	A 3-D sampled grid of the far-field pattern, with 720 samples

3.1.1 Generation of training data from an 8-element linear array

In this section, we describe how to generate the far-field pattern of each unique failure scenario and process the result so that it can be used to train a machine-learning algorithm.

A commercial computational electromagnetic (CEM) software tool, FEKO [48], was used to analyse and evaluate antenna designs. The three FEKO programs used in this work were CADFEKO, EDITFEKO and POSTFEKO. CADFEKO is a 3-D design program that was used to draw one element in the antenna array seen in Fig. 3.1. This definition of the antenna element geometry is stored as a mesh file. EDITFEKO is an editor that allows one to use the mesh file designed in CADFEKO, and describe the positions and on/off states of the antennas. The desired results such as the far-field pattern are also requested in EDITFEKO. Once the simulation has been executed, the results can be viewed in POSTFEKO. The far-field results were processed further by the machine-learning-based diagnostic utility developed in this work.

EDITFEKO uses the CADFEKO mesh file, along with an .xml file, to describe the problem that FEKO will solve. An .xml file is generated for each unique failure scenario, and contains the description of each antenna array, including its position and magnitude. The magnitude of an antenna is equal to 1 when the antenna is fully functional, and equal to 0 when the antenna has failed. Each unique scenario is given a scenario identification number, and a unique code that describes the ON or OFF state of the antenna. The different far-field patterns necessary to form the training data sets are requested in the EDITFEKO script.

Once the problem has been simulated, an output file is generated for each far-field request (as mentioned in Table 3.1). These files are in a format readable by POSTFEKO. At each sampling point, the real and imaginary $\vec{E}$-fields are measured in both the θ and ϕ directions. However, for the data to be useful for training, the output file for each failure scenario has to be processed before the files can be combined into one training data set. Each output file is processed so that the total absolute magnitude can be calculated from the measured results. This allows the training data to be compressed from a matrix of N x 4 to an array of N entries, where N is the number of samples taken of the far-field.

In other words, the far-field data contained in the output files generated after simulation are extracted. The data come in the form of the real and imaginary $\vec{E}$-field in both the θ and ϕ directions, i.e., 4 measurements per sampling point. For each sampling point, this is reduced to one value - the total magnitude for that point. After each sampling point has been processed, the whole pattern is normalised and converted to the logarithmic scale (dB). This is done because the small differences in low dB values are more prominent when expressed on a logarithmic scale, which makes it easier for the machine-learning algorithm to distinguish between different scenarios.

Once all the scenarios have been processed, the arrays are stacked together to form a training data set. Each failure scenario is labelled by its identification number. Each training data set has two matrices - the training data and the related labels. When the far-field data of each unique failure scenario is added to the training data set, the appropriate scenario number is saved in another array, at the same index. This index is called the label of the training data.

3.1.2 Other configurations and data sets

For more complicated antenna arrays, with more antennas, simulating all possible failure scenarios becomes too computationally expensive. Not all failure scenarios are going to occur. At some point, it is unlikely that many failures will occur at once, without being noticed by the system health management.

The purpose of this project was to find ways of detecting a few failed elements in a large array. So, for the sake of processing time and realism, as the number of elements in an array gets larger, it is good practice to simulate only up to a certain number of simultaneous failures. The number of unique failure scenarios R for an antenna array of n elements, with up to k simultaneous failures, can be calculated as:

$$R = \sum_{0}^{k} \binom{n}{k}. \tag{3.2}$$

Three array configurations were studied. The simple example of an 8-element linear antenna array can be extended to any arbitrary array configuration. The other two array configurations used in this work were a 25-element regular dense antenna array and an 18-element irregular sparse antenna array. These two configurations can be seen in Fig. 4.7.

Instead of having only one label per entry in the training data set, there can be multiple labels per entry. This is a machine-learning concept called a multi-label data set. For each entry in the training data set, there is an array of labels that describe its categories. In this case, the labels of each scenario reflect the ON or OFF state of each antenna for each failure scenario. An 8-element antenna would have 8 labels per entry in the training data set - each combination of labels being unique. Using this multi-label data, the machine-learning algorithm attempts to learn, by looking at the far-field pattern, whether each individual antenna in the array is ON or OFF.

3.2 Machine-learning Algorithms

In this section, we describe the different machine-learning algorithms used to train on the far-field data. We explain what machine learning is and how it works. We then describe the two main machine-learning algorithms used in this work - the FNN and the SVM.

3.2.1 Basic machine-learning concepts

In a general sense, machine learning corresponds to pattern recognition. A supervised machine-learning algorithm attempts to predict the label of a new, unknown example, based on observations of previous

data-label pairs that the algorithm has processed.

Machine learning has three phases: training, validation and testing[59], [60]. Consequently, the data set is usually divided into three sets - the training set, the validation set and the test set (usually 80%, 10% and 10% of the data respectively).

During the **training phase**, the training data is fed to the machine-learning algorithm. Each machine-learning algorithm's training phase is different, but the purpose of the training phase is for the machine-learning algorithm to form a picture of how the classes in a data set differ from one another. The machine-learning algorithm must be able to predict the class of a new example during the validation or test phase, based on knowledge learnt from the training phase. The SVM, for example, attempts to define the separation between classes in a multi-dimensional space. The FNN attempts to form a functional mapping of the training data and associated labels, using hyperparameters[59].

During the **validation phase**, the machine-learning algorithm's ability is assessed. The function that describes the mapping between the data inputs and consequent labels is no longer altered. The ability of the machine-learning algorithm to classify unseen data is validated. After using the validation set, one can go back and change the hyperparameters of the machine-learning algorithm, and retrain it on the training data set to attempt to achieve a better accuracy or training time. In other words, the validation data set is used to test and tune the machine-learning algorithm. The testing phase only starts once the machine-learning algorithm has been fine-tuned.

During the **testing phase**, the test set is used only once, to determine the accuracy of the algorithm. This is necessary because a machine-learning algorithm that does not generalise well enough can be over-trained in both the training and the validation phase. The test set gives a better idea of the true performance of the algorithm on unseen data if it is only used once.

3.3 Support Vector Models

SVMs were invented around 1979 by Vladimir Vapnik, a Russian scientist. After moving to the United States, he published his first paper on SVMs in 1995 [12].

SVMs are designed to split two classes of data so that the distance between the line of separation and the two classes is maximised. An SVM is visualised in Fig. 3.2. Two classes, denoted by + and -, are plotted on a 2-dimensional plane. The plane that separates the two classes is called the **hyperplane**. The **support vectors** are the data points from each class closest to the hyperplane. The separation zone between the support vectors of the two classes is called the **margin**. The SVM iteratively changes the orientation of the hyperplane until the margin is maximised. The maximisation of the margin is a constrained optimisation problem, which can be solved by using Lagrange multipliers [61].

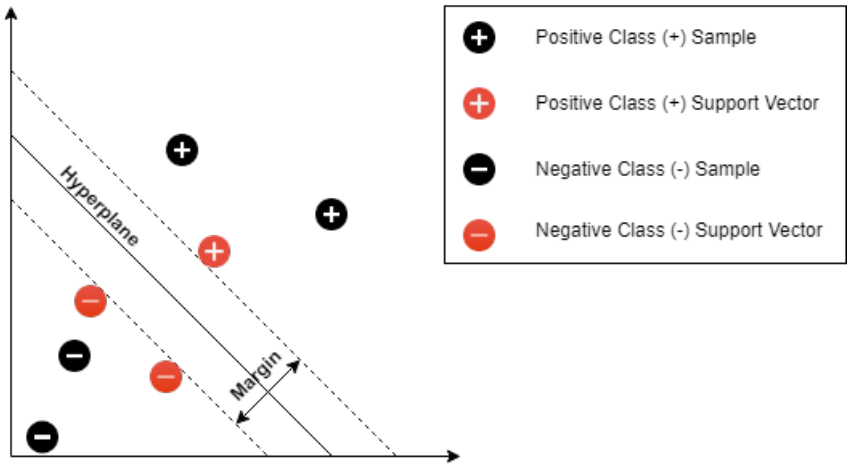


FIGURE 3.2: Demonstration of the SVM hyperplane and the margin that separates two classes. The support vectors of each class are shown in red.

In this section, we discuss the mathematics behind SVMs, starting with an SVM for linearly separable data. We explain how SVMs can be used on classes that do not seem linearly separable, and we introduce multi-class, multi-label data sets. The resources and textbooks used to write this section are *Introduction to Support Vector Machines* [62], *An introduction to support vector machines and other kernel-based learning methods* [63] and *Advances in kernel methods: support vector learning* [64].

3.3.1 Linear SVMs

Consider a set $(\mathbf{X}, Y)$ of l training examples $\{\mathbf{x}_i, y_i\}, i = 1, \ldots, l$. Each training example $\mathbf{x}_i$ has d features, so that the datapoints exist in a d-dimensional feature space $(\mathbf{x}_i \in \mathbb{R}^d)$. Each training example belongs to one of two classes, $(C_+$ or $C_-)$, so that the class label y_i of each training example can have one of two values, $(y_i \in \{-1, 1\})$.

All hyperplanes in the d-dimensional space $\mathbb{R}^d$ can be defined by

$$\{\mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = 0\}, \tag{3.3}$$

where b is a constant, and $\mathbf{w}$ is the vector passing through the origin, which is orthogonal to the hyperplane.

Consider such a hyperplane $(\mathbf{w}, b)$ that perfectly separates the classes C_+ and C_-. The function that correctly classifies the training data is then

$$f(\mathbf{x}) = \text{sign}\,(\mathbf{w} \cdot \mathbf{x} + b). \tag{3.4}$$

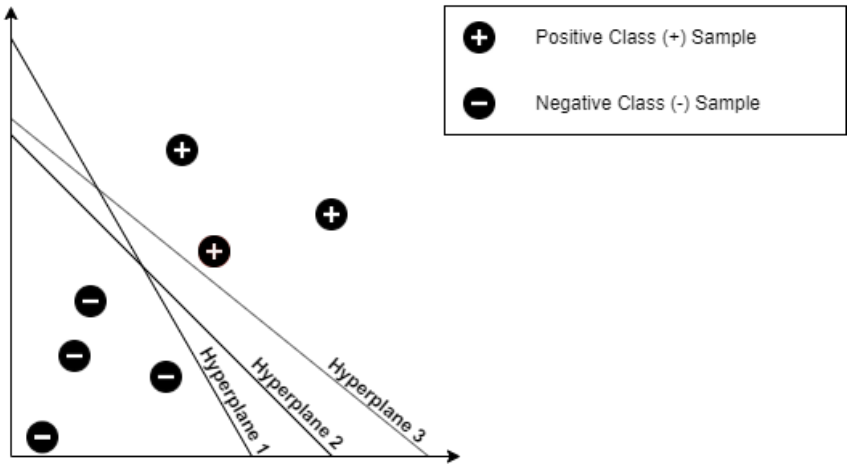


FIGURE 3.3: Many hyperplanes can separate classes C_+ and C_-.

From all the possible hyperplanes (see Fig. 3.3), the SVM should choose the hyperplane that separates the two classes with the widest possible margin. For this study, a canonical hyperplane [64] was chosen to separate the two classes. A hyperplane is in canonical form with regard to the training data $\mathbf{X}$ if it is a functional distance of at least one unit away from the closest data point of each class. The functional distance of a hyperplane is the value that the function equates to. This is not the euclidean or geometric distance, which is the margin. The hyperplanes that satisfy this requirement are

$$\mathbf{x_i} \cdot \mathbf{w} + b \geq +1 \quad \text{when} \quad y_i = +1, \quad \text{and} \tag{3.5}$$

$$\mathbf{x_i} \cdot \mathbf{w} + b \geq -1 \quad \text{when} \quad y_i = -1, \tag{3.6}$$

which can be condensed to

$$y_i \left(\mathbf{x_i} \cdot \mathbf{w} + b\right) \geq 1 \quad \forall_i. \tag{3.7}$$

In the case where the classes are completely seperable, this constraint can be met by the SVM. When the classes are non-seperable, the options are to use non-linear SVMs, discussed in Section 3.3.2, or introduce "slack variables", that allow the linear SVM to misclassify a few examples.

To calculate the geometric distance between a data point and the hyperplane, equation 3.7 is normalised using the magnitude of $\mathbf{w}$, which results in

$$d\left((\mathbf{w}, b), \mathbf{x_i}\right) = \frac{y_i \left(\mathbf{x_i} \cdot \mathbf{w} + b\right)}{||\mathbf{w}||} \geq \frac{2}{||\mathbf{w}||}. \tag{3.8}$$

From equation 3.8, one must minimise $||\mathbf{w}||$ to maximise the margin. We now solve a quadratic programming problem of optimization, also called a primal problem,

$$\min_{\mathbf{w},x} \frac{1}{2} ||\mathbf{w}||^2 ; \quad \text{subject to equation 3.7} \tag{3.9}$$

This problem can be rewritten using Lagrange multipliers [12], [65]. Equation 3.8 and equation 3.9 are parallel constraints that can be solved by recasting the constraints as a Lagrangian function by introducing new "slack variables", denoted α and requiring the derivative of the Lagrange function be zero. For a vector α of non-negative Lagrange multipliers, with a length l, the Lagrangian is given by

$$W(\mathbf{w}, b, \alpha) = \frac{1}{2} ||\mathbf{w}||^2 + \sum_{i=0}^{l} \alpha_i \left[y_i \left(\mathbf{x_i} \cdot \mathbf{w} - b \right) - 1 \right]. \tag{3.10}$$

Setting the derivative of the Lagrange function subject to $\mathbf{w}$ equal to zero results in the optimal hyperplane in equation 3.11, which shows that the vector $\mathbf{w}$ is just a linear, scaled sum of the training examples. Setting the derivative of the Lagrange function subject to b equal to zero results in equation 3.12, which will be used as a constraint for the Lagrangian function.

$$\frac{\partial W}{\partial \mathbf{w}} \implies \mathbf{w} - \sum_{i=0}^{l} \alpha_i y_i \mathbf{x_i} = 0 \implies \mathbf{w} = \sum_{i=0}^{l} \alpha_i y_i \mathbf{x_i}. \tag{3.11}$$

$$\frac{\partial W}{\partial b} \implies \sum_{i=0}^{l} \alpha_i y_i = 0 \tag{3.12}$$

Substituting the result of equation 3.11 into equation 3.10, and knowing the result of equation 3.12, we arrive at the minimised Lagrangian

$$W(\alpha) = -\sum_{i=0}^{l} \alpha_i + \frac{1}{2} \sum_{i=0}^{l} \sum_{j=0}^{l} y_i y_j \alpha_i \alpha_j \left(\mathbf{x_i} \cdot \mathbf{x_j} \right), \tag{3.13}$$

which is subject to the constraints below. The constant C is a tunable parameter that will be discussed soon.

$$\sum_{i=0}^{l} y_i \alpha_i = 0, \tag{3.14}$$

$$0 \leq \alpha_i \leq C \quad \forall_i. \tag{3.15}$$

Combining 3.7 and 3.11 with some manipulation, the equation

$$\alpha_i \left(y_i \left(\mathbf{x_i} \cdot \mathbf{w} + b \right) - 1 \right) = 0 \quad \forall_i, \tag{3.16}$$

shows that, when the distance between the hyperplane and a given example is larger than one, $\alpha_i = 0$, the example has no effect on the definition of the optimal hyperplane. This equation explains the name of the SVM - only the examples closest to the optimal hyperplane, with values $\alpha_i > 0$, have an effect on its position, and are therefore called the support vectors.

Equation 3.3 shows that two parameters are necessary to define the hyperplane, $\mathbf{w}$, which has been determined, and b. The parameter b can be calculated by taking any support vector from each class, to produce an $\mathbf{x}^+$ and an $\mathbf{x}^-$. One can solve the equations simultaneously (from equation 3.16),

$$\left(\mathbf{x}^+ \cdot \mathbf{w} + b \right) = +1 \quad \text{and} \tag{3.17}$$

$$\left(\mathbf{x}^- \cdot \mathbf{w} + b \right) = -1, \tag{3.18}$$

to yield

$$b = -\frac{1}{2} \left(\mathbf{x}^+ \cdot \mathbf{w} + \mathbf{x}^- \cdot \mathbf{w} \right). \tag{3.19}$$

Finally, the constraint C is a tunable parameter that determines the flexibility of the hyperplane. For large values of C, the hyperplane is very strict on classifying each example correctly. A lower C value allows the hyperplane to misclassify some examples, for the benefit of having a more generalised SVM. A more flexible hyperplane is useful for data that is not easily separable.

3.3.2 Non-linear SVMs

SVMs are not limited to linearly separable data. Using a kernel allows one to separate the training data linearly with a hyperplane in a higher dimensional feature space. A mapping defined as

$$\mathbf{z} = \phi \left(\mathbf{x} \right), \tag{3.20}$$

is used to transform a vector $\mathbf{x}$, of dimension d, into a vector $\mathbf{z}$, which has a higher dimensionality than d. The mapping function $\phi\left(\right)$ transforms the data into a higher dimension, where the data may be linearly separable by a hyperplane. Substituting $\mathbf{x}$ with $\phi\left(\mathbf{x}\right)$ in equation 3.13, we find

$$W(\alpha) = -\sum_{i=0}^{l} \alpha_i + \frac{1}{2} \sum_{i=0}^{l} \sum_{j=0}^{l} y_i y_j \alpha_i \alpha_j \left(\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j}) \right), \tag{3.21}$$

so that the equation describing the optimal hyperplane (**??**), becomes

$$\mathbf{w} = \sum_i \alpha_i y_i \, \phi(\mathbf{x_i}). \tag{3.22}$$

and, the function that correctly classifies the training data, from equation 3.4, becomes

$$\begin{aligned} f(\mathbf{x}) &= \text{sign}\left(\mathbf{w} \cdot \phi(\mathbf{x}) + b\right) \\ &= \text{sign}\left(\left[\sum_i \alpha_i y_i \, \phi(\mathbf{x_i})\right] \cdot \phi(\mathbf{x}) + b\right) \\ &= \text{sign}\left(\sum_i \alpha_i y_i \, \left(\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x})\right) + b\right). \end{aligned} \tag{3.23}$$

In each of these functions, it is important to notice that each $\phi(\mathbf{x_a})$ that appears is always in a dot product with another $\phi(\mathbf{x_b})$. We define this function as

$$K(\mathbf{x_a}, \mathbf{x_b}) = \phi(\mathbf{x_a}) \cdot \phi(\mathbf{x_b}). \tag{3.24}$$

Equation 3.24 is called the kernel [63]. The kernel K is a formula for the dot product of two vectors, $\mathbf{x_a}$ and $\mathbf{x_b}$, in a higher dimensional space. If there is a formula for the dot product of these two vectors in a higher dimensional space, it is no longer necessary to project the input vectors themselves into the higher dimensional space. It is not even necessary to know the formula of the mapping $\mathbf{z} = \phi(\mathbf{x})$. In other words, the optimal hyperplane will be in an unknown feature space, but the kernel, the formula for the dot product in that feature space, is all that needs to be known. The optimal hyperplane will appear as a curved or even non-continuous contour in the original feature space.

The classifier, from equation 3.4, can now be set up as

$$f(\mathbf{x}) = \text{sign}\left(\sum_i \alpha_i y_i \, \left(K(\mathbf{x_i}, \mathbf{x})\right) + b\right) \tag{3.25}$$

with an optimisation problem [61], [66], from equation 3.21, as

$$W(\alpha) = -\sum_{i=0}^{l} \alpha_i + \frac{1}{2} \sum_{i=0}^{l} \sum_{j=0}^{l} y_i y_j \alpha_i \alpha_j \left(K(\mathbf{x_i}, \mathbf{x_j})\right). \tag{3.26}$$

Using these equations, the optimal hyperplane can be found as described before. The only difference is that the hyperplane is in a feature space that is unknown.

To choose a kernel function K, the dot product between the two vectors must exist in a feature space for mapping $\phi()$. There are two methods to find a kernel function K that adheres to this requirement. The first, more tedious method, is to create a mapping $\mathbf{z} = \phi(\mathbf{x})$. from this mapping, the kernel $K(\mathbf{x_a}, \mathbf{x_b}) = \phi(\mathbf{x_a}) \cdot \phi(\mathbf{x_b})$ can be derived analytically. The second method is to choose a kernel function K, and check that it is a valid dot product in a higher dimensional feature space, using Mercer's condition [67]. The condition states that, for a choice of f_i's, a kernel function K can be written as

$$K(\mathbf{x_a}, \mathbf{x_b}) = \sum_{i=1}^{\infty} f_i(\mathbf{x_a}) f_i(\mathbf{x_b}). \tag{3.27}$$

The kernel function K is a dot product in a feature space if it satisfies the conditions

$$\begin{aligned} \int\int K(\mathbf{x_a}, \mathbf{x_b}) g(\mathbf{x_a}) g(\mathbf{x_b}) d\mathbf{x_a} d\mathbf{x_b} > 0 \\ \forall g \quad \text{such that} \\ 0 < \int g^2(\mathbf{x}) d\mathbf{x} < \infty. \end{aligned} \tag{3.28}$$

Two popular kernels that satisfy these requirements are the polynomial kernel, and the Gaussian radial basis function (RBF) [63], [66]. The polynomial kernel in equation 3.29 has a tunable parameter p, which is usually between 1 and 10.

$$K(\mathbf{x_a}, \mathbf{x_b}) = (\mathbf{x_a} \cdot \mathbf{x_b} + 1)^p \tag{3.29}$$

As the parameter p becomes larger, so the dimension of the feature space that the optimal hyperplane exists in becomes larger. The larger the feature space, the more likely it is that the data will be linearly separable. However, the more dimensions there are, the more support vectors are necessary to separate the two classes C_+ and C_-. This increases the complexity of the problem, which means that the computational complexity increases. Following Occam's razor [68], simpler systems are better. In the case of the polynomial kernel, one should use the lowest possible value for the parameter p, so that the data is represented in the most compact form (the lowest dimensional feature space). As the dimension increases, the more difficult it will become for the algorithm to generalise well [62].

The Gaussian RBF kernel [64], with a tunable parameter σ,

$$K\left(\mathbf{x_a}, \mathbf{x_b}\right) = e^{-\frac{||\mathbf{x_a}-\mathbf{x_b}||^2}{2\sigma^2}} \tag{3.30}$$

results in the classifier

$$f(\mathbf{x}) = \text{sign}\left(\sum_i \alpha_i y_i \, e^{-\frac{||\mathbf{x_a}-\mathbf{x_b}||^2}{2\sigma^2}} + b\right). \tag{3.31}$$

The shortcoming of SVMs is that their training time and memory complexity worsen as the size of the training set increases. For further reading, Burges [65] suggests ways to improve the accuracy and speed of SVMs, and Nalepa [69] describes existing ways to select training data for SVM training from large data sets.

3.4 Feedforward Neural Networks

3.4.1 From FNNs to SVMs

Linear models such as linear regression [70] or logistic regression [71] are reliable and efficient, but are limited to linear functions. This was overcome by the kernel trick introduced in section 3.3.2, where a mapping $\phi(\mathbf{x})$ is able to represent non-linear data in a higher dimensional space, where it is linearly separable. In other words, the linear function is applied, not to the non-linearly separable data $\mathbf{x}$, but rather to $\phi(\mathbf{x})$, which has been transformed to a space where it is indeed linearly separable. The mapping ϕ can be seen as a new representation for $\mathbf{x}$. In machine learning, the task is to choose the mapping ϕ.

SVMs use generic mappings ϕ, that have been proven useful for many applications. These mappings are implicitly used by using a kernel, as discussed in section 3.3.2. Another method is to manually engineer the mapping ϕ. This requires domain knowledge and specialists, and a new mapping must be designed for each new task. There is little transfer between domains, and engineering each mapping takes a lot of human effort and time. Deep learning's approach is to learn the mapping ϕ.

3.4.2 A broad overview of FNNs

A feedforward neural network (FNN) is the most basic and classic of the deep learning methods. They are used in many commercial and online applications. Some interesting applications of FNNs include autonomous driving [72], financial prediction [73], time series prediction [74] and text-to-phoneme

mapping [75]. FNNs are also commonly known as multilayer perceptrons (MLPs), and deep feedforward networks [59]. Other neural networks are built on the basis of FNNs. For example, a convolutional neural network is a special form of FNN, and is very powerful in object recognition in pictures. Recurrent neural networks are very good at natural language processing and video processing.

An FNN attempts to approximate some function

$$y = f^* (\mathbf{x}) , \tag{3.32}$$

where the classifier f^* maps the input $\mathbf{x}$ to the relevant class label y. This is the true relationship between the input $\mathbf{x}$ to the relevant class label y. The aim of the FNN is to approximate this relationship between $\mathbf{x}$ and y by defining a mapping

$$y' = f (\mathbf{x}, \mathbf{W}) , \tag{3.33}$$

where y' is the predicted class label and $\mathbf{W}$ is the learnable parameters that are updated iteratively, until the true relationship f^* is approximated as closely as possible, so that $y' \approx y$.

A neural network is called a network, because it is a combination of functions chained together. The feedforward neural network is the simplest type of neural network, and serves as a conceptual stepping stone for more complicated neural networks. It is called feedforward (as opposed to, for example, recurrent), because the output of one function is the input of the next. The network does not pass information back to previous functions. Networks with such feedback connections are called recurrent neural networks [59]. For example, if the three functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ are chained together to form a feedforward neural network, the function f would be

$$f (\mathbf{x}) = f^{(3)} \left(f^{(2)} \left(f^{(1)} (\mathbf{x}) \right) \right) . \tag{3.34}$$

The input vector $\mathbf{x}$ is fed to the first function, $f^{(1)}$, which is called the first layer in the FNN. The output of the first layer is passed to the second layer, $f^{(2)}$, and so forth. The depth of the model is the number of layers the model has. A neural network can have many layers, which is why it is called deep learning. The last layer, in this case $f^{(3)}$, is called the output layer. This final layer of the neural network should produce a value as close as possible to the label y, associated with the current training example $\mathbf{x}$.

Each training pair $(\mathbf{x}, y)$ can be seen as a noisy, approximate sample of the true function $f^* (\mathbf{x})$ such that

$$y \approx f^* (\mathbf{x}) . \tag{3.35}$$

As mentioned before, the neural network trained iteratively until $f(\mathbf{x})$ resembles $f^*(\mathbf{x})$ as closely as possible. This is achieved by feeding the FNN with noisy samples $\mathbf{x}$ and specifying that the output layer should produce the associated output y. All the layers in the FNN between the input layer and the output layer are called hidden layers, because there is no specification as to what the desired outputs of these layers should be. The learning algorithm must use these hidden layers to produce the desired output, an approximation of $f^*(\mathbf{x})$ in the final layer.

The neural network is called neural, because it was inspired by early understanding of neurons or perceptrons [59]. Each hidden layer $f^{(i)}$ in the neural network has several units of vector-to-scalar functions, called neurons, which act in parallel. Each neuron in the layer observes the output vector of the previous layer, and produces a scalar value. The scalar values produced by each neuron in the layer are combined to form the output vector of that layer. The number of neurons in the layer is the width of the layer. The idea of using many such layers is inspired by neuroscience, and the observation of how biological neurons work. However, the goal of an FNN isn't to design a perfect model of the biological brain, but rather to use mathematical and engineering principles that are guided by insights from what we have learned of how the brain functions. FNNs approximate functions to achieve statistical generalisation.

3.4.3 A mathematical breakdown of FNNs

The work in this section relies heavily on a discussion of feed-forward neural networks in chapter 20 of Artificial Intelligence - A Modern Approach by Russel and Norvig [76].

The smallest component of a neural network is a neuron. This idea of a machine-learning model designed as a network of neurons layered after one another is modelled after current theory on how the biological brain works. In the biological brain, a neuron is a cell that observes electrical signals from previous neurons in the network. The neuron processes the input signals according to some function, and emits a single electric response depending on a function of the input signals. Similarly, a neuron in a neural network will emit an output signal depending on a function of its inputs. The neuron will "fire" when the response of the function, a linear, weighted combination if its inputs, exceeds a certain value.

This operation of a neuron is very similar to **logistic regression**, which is used to map the relationship between a dependant output variable, given one or more independent input variables. Logistic regression is used for prediction of a binary output. Let the output variable a_i, be

$$a_i = g(\mathbf{x}_i) = g\left(\sum_{j=0}^{n} W_{j,i} a_j\right). \tag{3.36}$$

In the first step of logistic regression, the weighted sum of all inputs are calculated as $\sum_{j=0}^{n} W_{j,i}a_j$. We then require the output to be a probability – a value between 0 and 1. A non-linear function g is applied to this sum, to restrict the range of the output to [0,1]. The function most commonly used is the logistic function, better known as the sigmoid function (see Fig. 3.5 A), from where logistic regression gets its name.

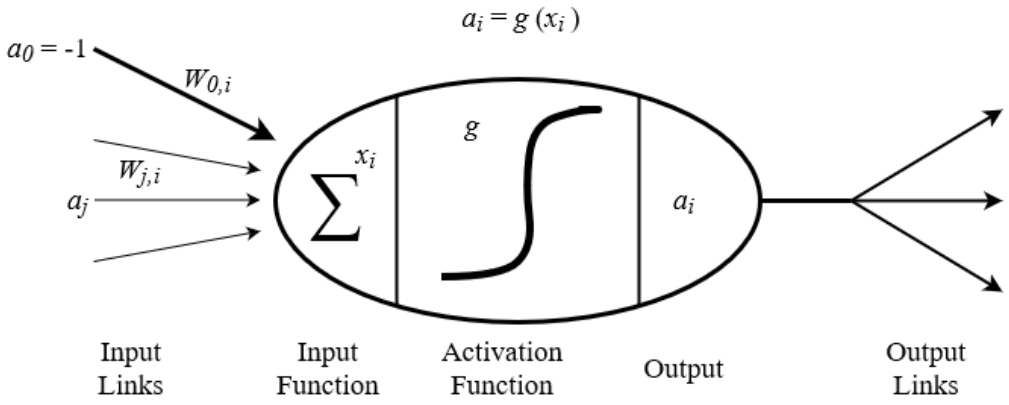


FIGURE 3.4: A visual demonstration of the mathematical model of a neuron devised by McCulloch and Pitts [77].

Fig. 3.4 illustrates the mathematical model of a neuron described by McCulloch and Pitts [77]. A neural network is composed of a series of neurons. One neuron in a neural network belongs to a layer, and is connected to the neurons in the previous layer, its inputs, and neurons in the next layer. The output of the neuron under discussion is connected to many other neurons in the next layer. These connections between neurons are called links. Consider two neurons, a neuron j in layer $l-1$, and a neuron i in layer l. An activation a_j is propagated through a link from neuron j to the current neuron i. Along with the activation a_j, each link also has a numeric, trainable weight $W_{j,i}$, which is a variable that controls how much effect the signal a_j has on the neuron i's output. The weight $W_{j,i}$ has a sign and intensity. A weighted sum of all the inputs to the neuron i is computed, so that

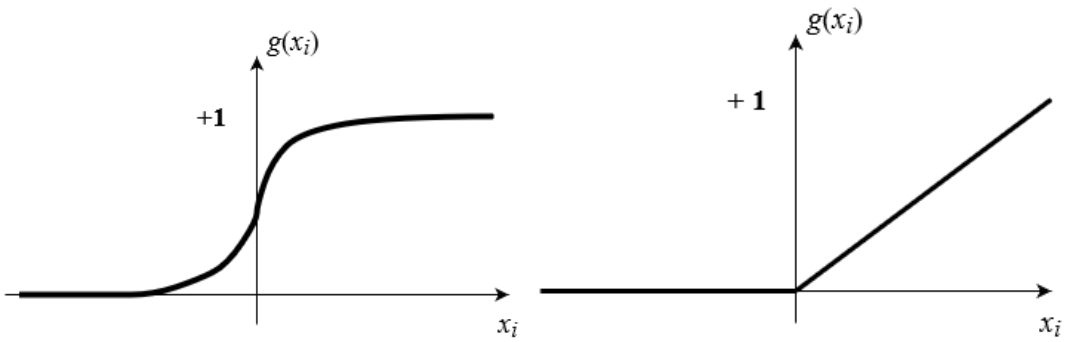
$$\mathbf{x}_i = \sum_{j=0}^{n} W_{j,i}a_j. \tag{3.37}$$

The activation a_i, the output of neuron i, is calculated by applying an activation function g to the weighted sum $\mathbf{x}_i$ calculated in equation 3.37,

Note that, in Fig. 3.4, there also exists a bias weight $W_{0,i}$. The weight is applied to a fixed input a_0 with a constant value of -1. This will be discussed soon.

There are two phenomenon that should be taken into account regarding the activation function g. First, the activation function of a neuron should generally be non-linear, as a linear function at

each neuron would cause the whole neural network to collapse into one linear function. Second, in many cases the neuron should emit a value close to $+1$ when the correct set of inputs are observed. This is referred to as "activating" the neuron. The neuron should emit a value near 0 otherwise. Typical activation functions are the sigmoid function, and the ReLU (rectified linear unit) function [78]. The graphs for these functions can be seen in Fig. 3.5.



(A) Sigmoid (B) ReLU

FIGURE 3.5: Two typical activation functions currently used for FNNs - the sigmoid function and the ReLU function.

The sigmoid function is differentiable, a useful property for learning the weights of the FNN. The sigmoid function has the form

$$g\left(x\right) = \frac{1}{1+e^{-x}}. \tag{3.38}$$

The ReLU function is the most commonly used activation function in neural networks, especially in convolutional neural networks [59]. Mathematically, the ReLU function is defined as

$$\begin{aligned} g\left(x\right) &= 0 \quad \text{for} \quad x \leq 0, \quad \text{and} \\ g\left(x\right) &= x \quad \text{otherwise.} \end{aligned} \tag{3.39}$$

The definitions of the activation functions have a threshold at $\mathbf{x}_i = 0$. In the neuron, the true threshold is set by the trainable bias weight, $W_{0,i}$. From equation 3.37, this means that

$$\begin{aligned} \mathbf{x}_i &= \sum_{j=0}^{n} W_{j,i} a_j \\ &= \sum_{j=1}^{n} W_{j,i} a_j - W_{0,i}, \end{aligned} \tag{3.40}$$

so that, when the first term, $\sum_{j=1}^{n} W_{j,i}a_j$ exceeds the value of $W_{0,i}$, the neuron is activated.

As said before, a neural network models its output as a function of its inputs. The more layers, the more complex the function that the neural network can represent. The network computes a function $y = f(\mathbf{x}, \mathbf{W})$, by adjusting the weights $\mathbf{W}$. The function the network represents can be changed by changing the values of these parameters. The network is trained by gradually adjusting these weights until the network produces the correct output for a known input.

Consider a simple case of a neural network discerning between only two classes, C_+ and C_-. The network would only have one output neuron. This neuron will have a threshold value of 0.5, so that an output with a value above 0.5 will be interpreted as C_+, and an output with a value below 0.5 will be interpreted as C_-. However, for a more complex neural network with k classes, it is more practical to have k output neurons. This way, the output of each neuron can represent the likelihood of the input $\mathbf{x}$ belonging to that class.

3.4.4 Multiple layers

The depth of the neural network - the number of hidden layers - determines the complexity of the function that can be represented. The width of such a hidden layer, which is the number of neurons per layer, depends on the length of the input vector. Interestingly, it is possible to prove that any continuous function of the input vector can be approximated by only one hidden layer, given that its width is sufficient. With two hidden layers, it is possible to represent discontinuous functions. The proofs are complex, but it is possible to prove that the required width for a hidden layer increases exponentially as a function of the length of the input vector [76]. However, it is very difficult to characterise exactly which types of functions can be represented, given a particular network structure. It is still not well understood how to choose the number of layers and width of each layer given a particular problem.

Consider a problem (see Fig. 3.6) where the training data set has a matrix of training examples $\mathbf{X}$ and labels $\mathbf{y}$ so that $\mathbf{X}$ contains n training examples $\mathbf{x}$, each with an associated label y. The input vector will be the size of one training example. The first layer will therefore have 10 neurons - one to observe each element of the input vector. For this problem, we will choose one hidden layer with a width of 4 neurons, and an output layer with one neuron. However, for problems with several categories, it is more common to have one output neuron per category, so that the final layer represents the probability of the current example $\mathbf{x}$ belonging to each category.

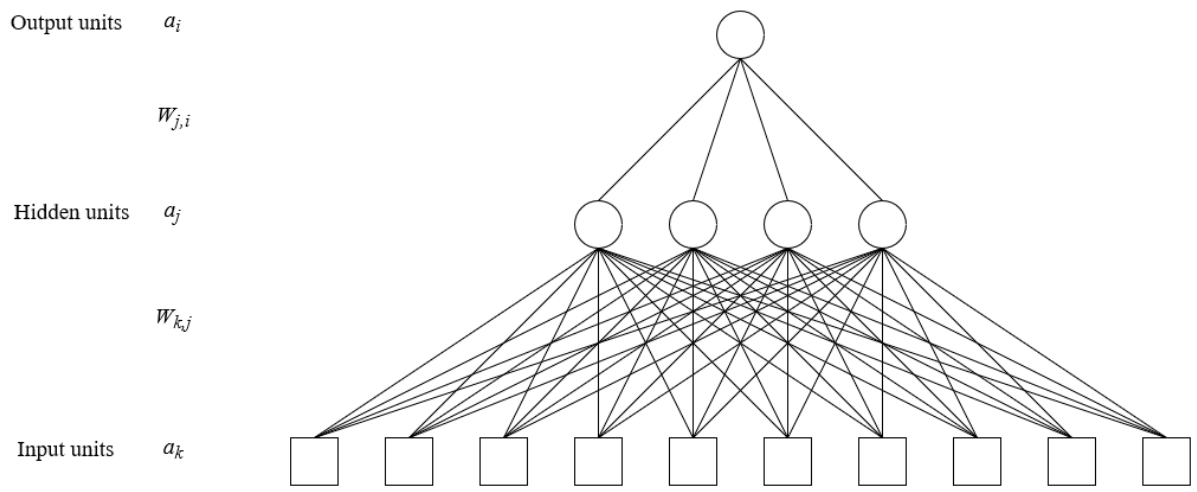


FIGURE 3.6: A neural network with 10 input neurons, 1 hidden layer with a width of 4, and a single output neuron.

3.4.5 Back-propagation

For each training example **x**, the neural network produces an output vector **y'** $= f(\mathbf{x}, \mathbf{W})$, the predicted output vector for the observed input vector. This predicted output **y'** is compared to the expected output vector **y**, as specified by the training data set. Both **y** and **y'** can be vectors, where each element of the output vector is the probability of the observed example **x** being in that class. A calculation, max (**y'**), produces the winning probability which refers to the (scalar) class label y' in equation 3.33.

The training of a neural network has two stages: the forward pass and the back-propagation of the error. During the forward pass, the neural network predicts an output vector **y'** for a given training example **x**. Next, the error vector **Err**, the error between **y'** and **y**, is calculated. This error is back-propagated through the hidden layers, using gradient descent. Let the error vector be defined as

$$\mathbf{Err} = \mathbf{y} - \mathbf{y'} = \mathbf{y} - f(\mathbf{x}, \mathbf{W}) . \tag{3.41}$$

The reason why the error is back-propagated, is to adjust the weights **W** of the network, so that the error **Err** is minimised. A classic error measurement is the sum of squared errors. This error is defined as

$$\mathbf{E} = \frac{1}{2}\mathbf{Err}^2 \tag{3.42}$$

The squared error for a single training example is a sum over the outputs y_i of each neuron in the output layer. We define this error E as

$$E = \frac{1}{2} \sum_i (y_i - a_i)^2 . \tag{3.43}$$

Now, to get the gradient for a specific weight $W_{j,i}$, the derivative of E in equation 3.43 is taken with respect to each weight $W_{j,i}$. The derivative is written in terms of the activation a_i. All other terms in the summation in equation 3.43 are unaffected by the derivation over $W_{j,i}$.

Gradient descent [78] is used for back-propagation. With gradient descent, the aim is to calculate the partial derivative of **E** with respect to each weight. First, let Err_i be the i'th element of **Err**. In what follows, g' is the partial derivative of the activation function, and α is the learning rate. The learning rate is a parameter chosen for the whole network, before training. Through manipulation, we can prove that the weight-update rule, based on the gradient descent algorithm, can be written as

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times Err_i \times g'(\mathbf{x}_i) . \tag{3.44}$$

For convenience later, we define a modified error Δ_i so that

$$\Delta_i = Err_i \times g'(\mathbf{x}_i) , \tag{3.45}$$

and equation 3.44 becomes

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i . \tag{3.46}$$

Consider the neuron in Fig. 3.4, as a component of the neural network shown in Fig. 3.6. The function that describes the link between the neuron i and the neuron j in the previous layer, are subject to a weight $W_{j,i}$ and an activation signal a_j. After the error Δ_i has been calculated for a neuron in the output layer, using equation 3.45, and the weight of that link has been updated, as shown in equation 3.46, it is time to update the weights of the links in the hidden layer. To update the weight $W_{j,i}$ in a hidden layer, we need to define a term Δ_j analogous to the error term Δ_i that was used for the output nodes.

This is where the error back-propagation method is used. The concept is that one of the neurons j in the hidden layer $l-1$ has a partial effect on the error Δ_i for each neuron i in the output layer l. Each Δ_i value is divided according to the strength of the link between the output neuron i and each neuron j. The error term Δ_j for each neuron j is therefore a weighted sum of each error term Δ_i, and the activation value a_j for the link to each output neuron i, that were just calculated in the previous forward pass. This function can be written as

$$\Delta_j = g'(\mathbf{x}_j) \sum_i W_{j,i} \Delta_i. \tag{3.47}$$

Referring back to Fig. 3.6, the weight-update rule from the hidden layer to the input layer is almost identical to 3.46, with k counting the number of neurons in the input layer, and a_k the activation for a link between the input and hidden layer,

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j. \tag{3.48}$$

In summary, the back-propagation method works as follows:

- calculate the error Δ between the predicted and expected outputs, **y'** and **y**,
- for each layer, starting with the output layer and working through the hidden layers until you reach the input layer,
  - popagate the error values Δ back to the previous layer, as a function of the activation of each link (which was calculated in the forward pass),
  - update the weights W between the two layers.

In this chapter a review of the SVM and FNN algorithms was presented. In the following chapter the algorithms will be applied to various numerical examples.

4 Failure detection in simulated aperture arrays based on far-field sampling

This chapter describes two experiments to discover how the far-field radiation patterns of several different array configurations affect the accuracy of classification algorithms to identify failed elements in the array.

The far-field radiation pattern was sampled using different methods to determine which sampling method contained the most information. The antenna array used throughout to demonstrate concepts and generate results was the 5×5 bow-tie antenna array shown in Fig. 4.1.

When the 5×5 bow-tie antenna array in Fig. 4.1 is fully functional, the far-field pattern's geometric shape is symmetrical in the $x-z$ and $y-z$ planes, otherwise referred to as the $\phi = 0°$ cut and $\phi = 90°$ cut respectively. Previous work has shown that, when given the far-field pattern of an antenna array (usually the $\phi = 0°$ cut), a machine-learning algorithm, such as an FNN [38], [44] or an SVM [30], [40] , can predict failed elements in an antenna array.

To demonstrate why it is possible for classification algorithms to distinguish between failure scenarios, the $\phi = 0°$ cut and $\phi = 90°$ cut of the 5×5 bow-tie antenna array are shown in Fig. 4.2. Blue denotes the far-field pattern of the fully functional array and red denotes the far-field pattern of the array where elements 1 and 2 have failed. Fig. 4.2 demonstrates that, even at the first null, there is a 10 dB difference between the failed and the fully functional scenarios. Similarly, each unique failure

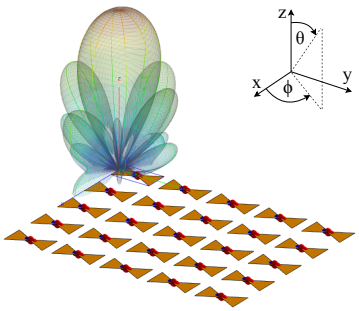


FIGURE 4.1: Three-dimensional far-field pattern of a fully functional regular 5×5 bow-tie antenna array, showing the θ and ϕ orientations.

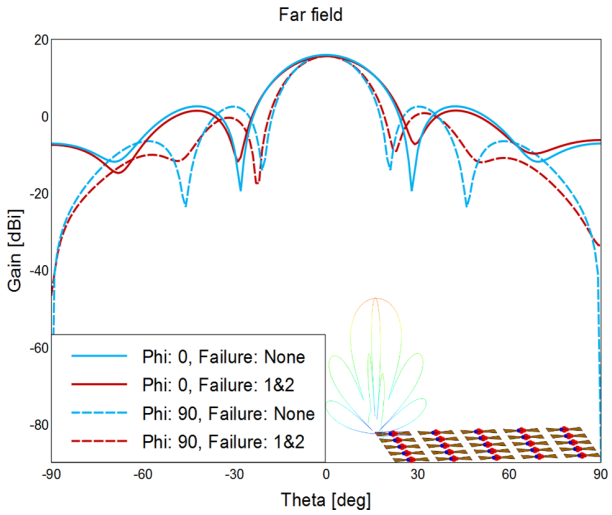


FIGURE 4.2: Far-field cuts of fully functional and partially failed 5×5 antenna arrays (see Fig. 4.1).

scenario has a unique three-dimensional (3-D) geometrical far-field pattern. This is why it is possible for a classification algorithm to distinguish between failure scenarios when given the far-field pattern as training data.

4.1 Overview

This chapter describes two experiments to investigate the effect of the choice of training data on a classification algorithm's accuracy. The two experiments are compared in Table 4.1.

Section 4.2 describes an experiment to discover how the far-field of a 5×5 bow-tie antenna array can be used to train an FNN and other classification algorithms. The classes of training data were equal to the number of antennas, similar to what was done by Patnaik [38]. The experiment had three objectives, the first of which was to determine which far-field sampling method to use to reduce the training data before training the classification algorithm. The results of the sampling methods on the FNN were compared for training time and accuracy. The second objective was to see how different classification algorithms performed with the different sampling methods, and the third objective was to determine which combination of classification algorithm and sampling method yielded the best results. The results of different algorithms were compared according to their accuracy in the different sampling methods.

Section 4.3 describes an experiment simulating three different array configurations to generate training data and investigate the methods introduced by Yeo [30]. An SVM was used as a classification model, and the number of classes was equal to the number of unique failure scenarios. The ideal

TABLE 4.1: A summary of the setup and objectives of two experiments in Chapter 4.

	Experiment 1	**Experiment 2**
Description	Investigation of the effect of sampling methods on the accuracy of different classification algorithms	Investigation of the effect of antenna configurations on the accuracy of the SVM kernel. Based on the method proposed by Yeo [30].
Classification Algorithms	An FNN*,[a]; and 14 out-of-the-box scikit-learn [45] classification algorithms.	An SVM[b] with a linear kernel*, second order polynomial kernel, and RBF kernel.
Training Data Sets	10 far-field sampling methods (see Table 4.2).	2 sampling methods: 3-D sampled far-field pattern, and the $\phi = 0°$ cut. The training data is taken from the ideal simulated patterns, and the test data is taken from noisy samples of the ideal pattern (see Fig. 4.6).
Array Configurations	25-element (5×5) regular dense bow-tie antenna array; up to 2 simultaneous failures (326 unique scenarios) (see Fig. 4.1).	25-element (5×5) regular dense bow-tie antenna array; up to 2 simultaneous failures (326 unique scenarios); 18-element irregular bow-tie antenna array; up to 3 simultaneous failures (988 unique scenarios); and 8-element linear bow-tie antenna array; up to 8 simultaneous failures (256 unique scenarios) (see Fig. 4.7).
Objectives	Determine which far-field sampling method to use to reduce the training data before training the classification algorithm. Compare the results of the sampling methods on the FNN for training time and accuracy. Evaluate how different classification algorithms performed with the different sampling methods, and Determine which combination of classification algorithm and sampling method yielded the best results. Compare the results of different algorithms for their accuracy in the different sampling methods.	Determine which SVM kernel achieved the highest accuracy and use this kernel throughout the rest of the experiment. Compare the results of different configurations to determine the effect of the number of elements, the number of classes and the layout of an array configuration on the accuracy of the SVM.
Notes	[a] In the training data set, the number of classes is equal to the number of *antennas* in the array, to allow the FNN to have more training examples per class.	[b] In the training data set, the number of classes is equal to the number of *scenarios* in the array, to follow the method proposed by Yeo [30].

* Primary classification algorithm.

far-field pattern of a failure scenario was used for the SVM training data and the same pattern with worsening SNRs was used for the test data. There were two objectives for this, the first of which was to determine which SVM kernel achieved the highest accuracy so that this kernel could be used throughout the rest of the experiment. The second objective was to compare the results of different configurations to determine the effect of the number of elements, number of classes and layout of an array configuration on the accuracy of the SVM. The results of the two experiments are discussed in Section 4.4.

4.2 Experiment 1: Sampling Methods

This section explains how an FNN and other classification algorithms were used to detect and locate failed antenna array elements in the 5×5 bow-tie antenna array shown in Fig. 4.1. The far-field pattern was used as input training data and different sampling methods were used on the original 3-D geometrical far-field pattern to generate training data sets for the FNN. The purpose of this experiment was to see which far-field sampling strategy yielded a training data set on which a given classification algorithm achieved the highest accuracy and shortest training time. In the first part of the experiment, the exact same FNN model was trained on ten different sampling method data sets. The accuracy and training times for each sampling method data set were compared to discover which sampling method resulted in the best information for the FNN to distinguish between failure scenarios. In the second part of the experiment, the ten different sampling methods were used to compare the performance of 14 out-of-the-box scikit-learn [45] multi-label classification algorithms to determine which types of classification algorithms were likely to perform well on the provided data.

4.2.1 Methodology

To generate the training data for the classification algorithm, all possible element failure scenarios were simulated for up to two simultaneous failures. The data generation process is demonstrated in Fig. 4.3. First, all possible failure scenarios were calculated and listed. The ON or OFF state of each antenna in a failure scenario was represented by 0 or 1 respectively. For each listed failure scenario, the appropriate antenna in the simulation model was turned ON or OFF. The resulting far-field was stored, and served as input for the machine-learning algorithms.

Each unique failure scenario served as a training example for the classification algorithm. After training on training examples, the classification algorithm attempted to predict the most likely scenario to have caused the observed far-field pattern. In doing so, the array excitation law could be modelled by the trained classification algorithm to indicate defective elements.

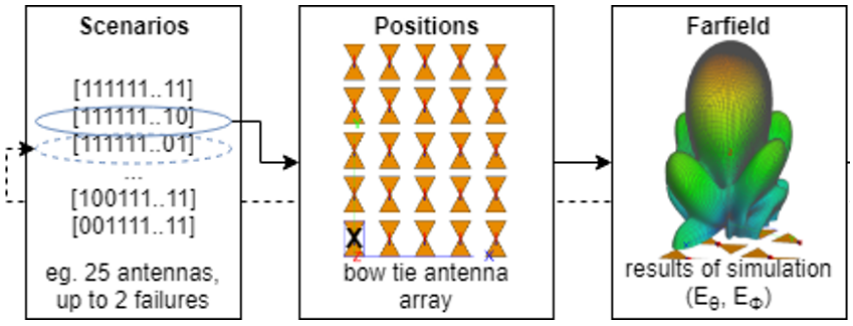


FIGURE 4.3: A demonstration of how the training data was generated.

Sampling methods

Ten different methods of sampling the simulated far-field pattern were compared as training data for the same FNN model.

- The first four sampling methods were slices in the 3-D plane at $\phi \in \{0°, 45°, 90°, 135°\}$ and samples were taken for the integer interval $\theta \in [-90°, 90°]$, resulting in 181 samples in each data set (refer to Fig. 4.1 for definitions of θ and ϕ).

- The next three sampling methods were combinations of the above-mentioned cuts. The *principle cuts* (the slices in the 3-D plane at $\phi = 0°$ and $\phi = 90°$) and *diagonal cuts* (the slices taken at $\phi = 45°$ and $\phi = 135°$) data sets had 362 samples each, so that the *all cuts* data set had 724 samples.

- The last three sampling methods were based on the 3-D far-field pattern in a (θ, ϕ) grid. The three methods are referred to as the *3-D pattern (182 samples)*, the *3-D pattern (361 samples)*, and the *3-D pattern (725 samples)*.

The sampling methods are summarised in Table 4.2 below.

Training of classification algorithm

For this experiment, the classification algorithms had to be able to predict multi-label data. Each input (x) was the sampled far-field observation of one failure scenario. For each observation, the labels (y) were the ON or OFF state of each antenna in the array. For the bow-tie array, there were 25 labels per scenario, i.e., one for each antenna element. The data set used to train the classification model was a 3-D matrix containing the far-field data of all possible scenarios.

The 3-D matrix had the dimensions $R \times S \times T$, where R was the number of scenarios, S was the number of sampling points, and T was the number of measurements taken at each sampling point. The number of scenarios R can be calculated as follows:

TABLE 4.2: Definitions of 10 sampling methods.

Sampling Method Name	**Number of Samples**	ϕ	θ
Single cut ($\phi = 0°$)	181	0°	$\theta \in [-90°, 90°]$
Single cut ($\phi = 45°$)	181	45°	$\theta \in [-90°, 90°]$
Single cut ($\phi = 90°$)	181	90°	$\theta \in [-90°, 90°]$
Single cut ($\phi = 135°$)	181	135°	$\theta \in [-90°, 90°]$
Principle cuts	362	0°, 90°	$\theta \in [-90°, 90°]$
Diagonal cuts	362	45°, 135°	$\theta \in [-90°, 90°]$
All cuts	724	0°, 45°, 90°, 135°	$\theta \in [-90°, 90°]$
3-D pattern (182 samples)	182	*	*
3-D pattern (361 samples)	361	*	*
3-D pattern (725 samples)	725	*	*

* 3-D far-field pattern sampled in a (θ, ϕ) grid.

$$R = \sum_{k=0}^{K} \binom{n}{k}, \tag{4.1}$$

where n is the number of antennas in the array and k is the number of antennas that could fail. For the 5×5 bow-tie antenna array, $n = 25$ and $k = 2$, resulting in 326 scenarios. The number of far-field sampling points, S, was determined by the sampling method. The four measurements, T, taken at each sampling point were $Re\{E_\phi\}$, $Im\{E_\phi\}$, $Re\{E_\theta\}$ and $Im\{E_\theta\}$.

4.2.2 Results

The accuracies achieved by the model for each data set are presented in Fig. 4.4. The four data sets resulting in the highest accuracies are compared according to their training times in Table 4.3. This table makes it clear that the training time was proportional to the number of samples.

Feedforward Neural Network

The results in Fig. 4.4 demonstrate the nature of the FNN, i.e., a drop in accuracy with an increase in the number of far-field samples. Data sets with more samples resulted in more network parameters Ψ. As the number of network parameters increased, more training iterations would be required to approximate the function f 3.32, than the current experimental setup allowed. Whenever a result deviated from this (the accuracy increased even though the number of samples stayed the same or even increased), it indicated that the sampling method had found areas in the far-field pattern that contained information useful for accurately identifying the failure scenarios. Examples include the difference in accuracy between *single cuts* and *3-D pattern (182 samples)*, the jump in accuracy from

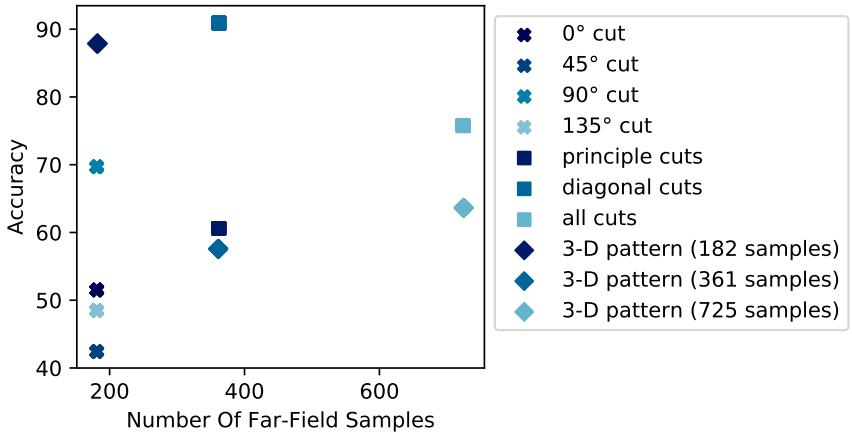


FIGURE 4.4: Accuracy vs number of far-field samples for each sampling method data set.

single cuts to *diagonal cuts*, and the jump in accuracy from *3-D pattern (361 samples)* to *3-D pattern (725 samples)*. The highest accuracies for the array configuration shown in Fig. 4.1 were achieved when sampling the far-field pattern using the *diagonal cuts* and *3-D pattern (182 samples)* methods.

TABLE 4.3: Training time of the four most accurate data sets.

Data set	Samples	Training Time (sec)	Accuracy (%)
90° cut	181	31.98	69.70
3-D pattern	182	32.17	87.88
Diagonal cuts	362	35.48	90.91
All cuts	724	40.73	75.76

Comparison of classification algorithms

This section explains how the sampling method data sets compared in the first part of the experiment were used to compare the accuracies of 14 out-of-the-box multi-label classification algorithms [45]. The algorithms and their average accuracies over all data sets are listed in Table 4.4 below.

The four best classification algorithms, according to their accuracy on the 10 sampling data sets, were the *FNN*, *One-vs-rest Classifier with Linear SVC core*, *One-vs-rest Classifier with Logistic Regression CV core* and *One-vs-rest Classifier with Logistic Regression core* classification algorithms. The accuracies of the four classifiers are plotted for each of the 10 sampling method data sets in Fig. 4.5 below. The algorithm that performed best overall was the *One-vs-rest Classifier with Logistic Regression CV core*, which is a one-vs-rest classifier with a logistic regression cross-validation core. This classification algorithm achieved a 100% accuracy on the *all cuts* data set, while the *One-vs-rest Classifier with Linear SVC core* and *One-vs-rest Classifier with Logistic Regression core* achieved 100% accuracy on the *3-D*

TABLE 4.4: Average accuracy of the four most accurate data sets.

Classification Algorithm	**Average Accuracy (%)**
Decision Tree Classifier	0
Extra Tree Classifier	0
Extra Trees Classifier	5.1515
Random Forest Classifier	4.5454
K Neighbors Classifier	2.1212
Radius Neighbors Classifier	0.606
One-vs-rest Classifier with Gradient Boosting Classifier core	31.5151
One-vs-rest Classifier with Gaussian Process Classifier core	0
One-vs-rest Classifier with Linear SVC core	66.6666*
One-vs-rest Classifier with Logistic Regression core	77.8787*
One-vs-rest Classifier with Logistic Regression CV core	78.1818*
One-vs-rest Classifier with SGD Classifier core	11.8181
One-vs-rest Classifier with Perceptron core	9.9999
One-vs-rest Classifier with Passive Aggressive Classifier core	18.7878
FNN	64.8484*

* The four classification algorithms that achieved the highest accuracy are indicated.

pattern (182 samples) data set. The *One-vs-rest Classifier with Logistic Regression core* achieved a better overall accuracy on the *combined cuts* and *3-D pattern* data sets, while the *One-vs-rest Classifier with Logistic Regression CV core* achieved a better overall accuracy on the *single cuts* data sets.

From Fig. 4.5, one can see that the 3-D sampling method datasets generally contained more discriminative information to distinguish between classes than the sampling methods based on *single cuts*. It was also clear that the other classification algorithms did not have the same relationship between the number of features and the accuracy of the algorithm that were seen for the FNN in the first part of Experiment 1. The FNN tends to have a drop in accuracy with an increase in the number of far-field samples. However, the results of the other three classification algorithms in Fig. 4.5 show an increase in accuracy as the number of samples increase. This behaviour is best demonstrated by the *One-vs-rest Classifier with Logistic Regression core*, that achieves the highest accuracy for the *Principal cuts*, *Diagonal cuts*, *3-D pattern (361 samples)* and *3-D pattern (725 samples)* data sets.

4.2.3 Conclusion

In this section, an FNN was used to detect and locate failed antenna elements in a bow-tie array and investigate the effect of the choice of training data on the machine-learning model's accuracy and training time. The results indicate that training the model on a data set using the *diagonal cuts* sampling method achieves the best accuracy for this antenna array (90.91% accuracy). However, the *3-D pattern*

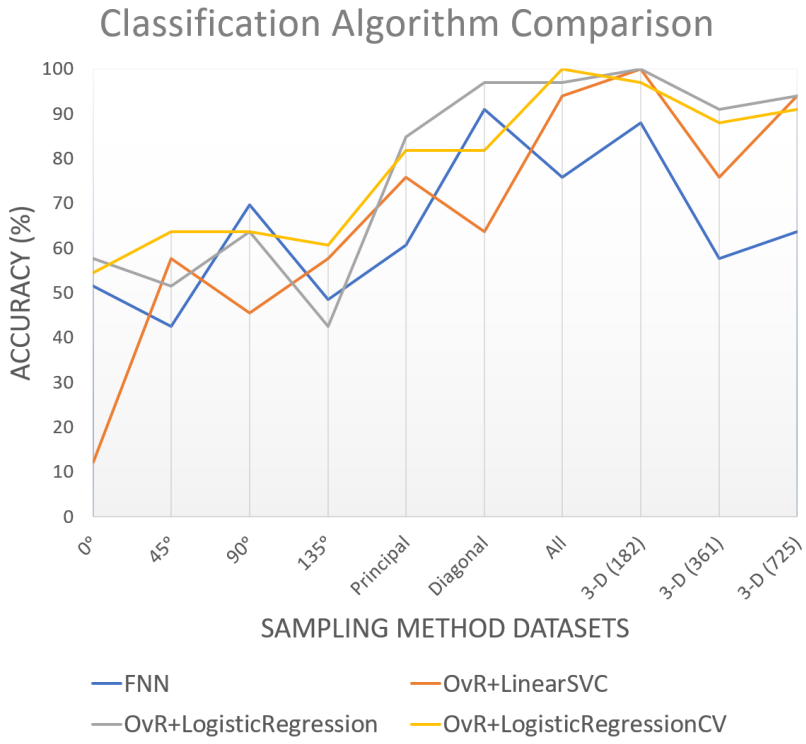


FIGURE 4.5: Accuracy of the four best classification algorithms on the 10 sampling method data sets

(182 samples) method achieves a shorter training time. If the experiment were repeated on a data set with more scenarios, e.g., a larger n or k (4.1), this difference in training time might become even more significant. Comparing out-of-the-box multi-label classification algorithms enabled us to postulate which classification algorithms were likely to do well on far-field data as training data. The *One-vs-rest Classifier with Logistic Regression CV core* achieved the best overall accuracy on the 10 sampling method data sets. From Fig. 4.5, it can be seen that the *3-D* sampling methods generally achieved a higher accuracy, which indicates that these sampling methods contain more information for distinguishing between failure scenarios.

4.3 Experiment 2: Array Configurations

The focus of this experiment is to investigate the possibility of using the SVM method introduced by Yeo [30] on large antenna arrays, such as those planned for the SKA radio astronomy project[7]. The experiment compared the accuracies of SVM kernels on increasing SNRs.

Methods similar to those described by Yeo [30] were used to identify failed elements, including

binary codification of the ON or OFF state of each antenna in the array in order to list all the unique failure scenarios, and the definition of classes as the scenario numbers. The array configurations shown in Fig. 4.7, were used in this experiment to train SVMs with different low-degree polynomial kernels.

The rest of this section reports which combination of kernels and sampling methods yielded the best results, and how the configuration of the antenna array affected the classifier accuracy.

4.3.1 Methodology

Representation and simulation of failure scenarios

Element failure can occur in a number of ways, as described by Yeo [30]. For this experiment, any element in a given array could be either fully functional (ON) or not radiating (OFF). To generate a far-field pattern for each unique failure scenario, faulty elements were turned completely OFF. To simulate all possible failure scenarios would have required 2^n combinations, where n is the number of elements in the given array. As n increases, so does the computational time required to simulate all unique failure scenarios, as well as to train the machine-learning algorithm on each possible scenario. As it is unlikely that a large number of failed elements would go undetected by the array's maintenance team, it was only considered necessary to design the machine-learning algorithm to identify a small number of failed elements that might otherwise be hard to find. The number of unique failure scenarios R for an antenna with n elements, and up to k element failures per scenario, was calculated in (4.1) in Section 4.2.

The method of presenting the state of all elements per unique failure scenario as a binary code, introduced by Yeo [30], is used in this work, as shown in Table 4.5. In the binary representation, 0 represents a fully functional element, and 1 indicates that the element has failed. A scenario identification number (SID) was assigned to each unique binary code. The binary representation is used to simulate the scenario identified by each SID, to generate an ideal far-field pattern for each SID.

Table 4.5: Representation of failure scenarios

SID*	Binary encoding	Description
1	0000...0000	All elements ON
2	0000...0001	n^{th} element OFF
...	...	...
R-1	1100...0000	Elements 1 and 2 OFF
R	1110...0000	Elements 1, 2 and 3 OFF

* SID: scenario identification number

Table 4.5 shows the scenario numbers and binary codes for an arbitrary array with n elements in a Cartesian grid. The number of elements per failure scenario (k) was chosen to be equal to 3, resulting

in a number of scenarios equal to R. Elements on the Cartesian grid were numbered first along the positive x axis, and then along the positive y axis, starting from the element closest to the origin. The most significant bit in the table shows the first element's state, and the least significant bit the n^{th} element's state.

Training and testing data generation

To train an SVM, the number of classes were equal to the number of scenarios (R). Each training example consists of a label (y) and feature vector (x). The labels of the data set were the SIDs of the failure scenarios, shown in the first column of Table 4.5. The feature vectors (x) of each training example were samples of the normalised amplitude (in decibels) of the far-field corresponding to the specific SID. The far-field pattern of each unique failure scenario was simulated using FEKO, to include the effects of mutual coupling of antennas in the array. When given a sampled far-field pattern (x), the trained SVM attempts to predict the label (y), from which a reference to a table such as Table 4.5 will identify which elements have failed.

In this work, each unique failure scenario was deemed a class, with the SID taken as the class name. To train the SVM model, the ideal far-field pattern of each scenario was fed to the SVM. The SVM thus received one perfect training example per class, and received no noisy far-field patterns during training. This method proved successful for SVMs [30], but will not work for other machine-learning algorithms such as FNNs, which require many training examples per class.

Once the SVM had been trained, its ability to predict the SID (y) from a given noisy far-field pattern (x) was tested. Increasingly noisy versions of each class were given to the classification algorithm as test data. The test set was produced by generating ten examples of what a given far-field pattern would look like at a certain SNR. Thus, 10 noisy examples of each of the R far-field patterns were generated for 12 SNRs, ranging between 25 dB and 0 dB. The size of the test set T was therefore

$$T = R \times 10 \times 12. \tag{4.2}$$

Fig. 4.6 shows an example of the training and testing data given to the SVM, and how the SNR affected the far-field pattern. The training data was the ideal far-field pattern, and test data of the same class became increasingly noisy, until it was impossible to recognise the original pattern.

4.3.2 Results

The aim of this section is to report what effect the choice of training data had on the reliability and accuracy of an SVM with a low-degree polynomial kernel. As the test examples became noisier, it

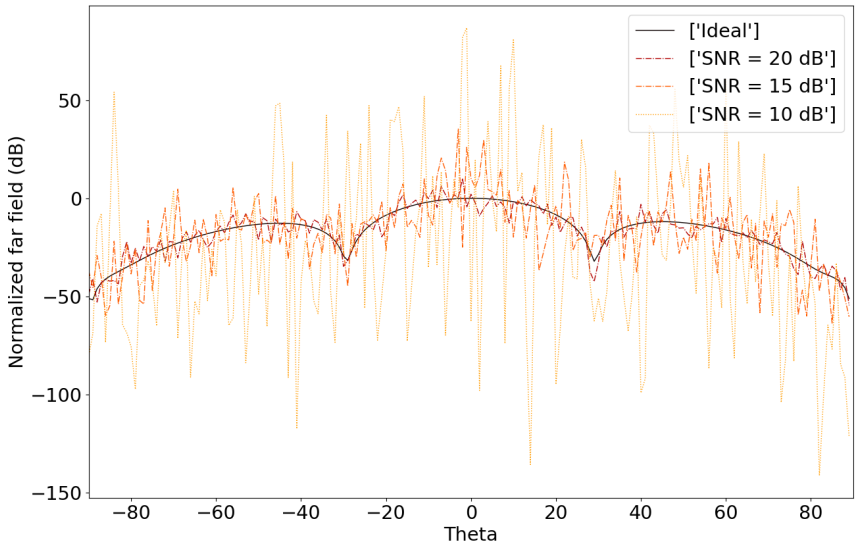


FIGURE 4.6: The ideal far-field pattern for scenario 1, used as training data for the SVM, is shown in black. 10 examples of the pattern were generated at varying levels of SNR to provide testing data. An example of testing data is shown in red for a SNR of 20, in orange for a SNR of 15, and in yellow for a SNR of 10.

became more difficult for the SVM to predict the class label (y). For this experiment, the reliability of the SVM lay in how accurately the SVM could classify a given noisy pattern for worsening SNRs. The data sets used to train the model were the far-field patterns of different array configurations with an increasing number of elements. The three configurations used are shown in Fig. 4.7. For each configuration, the far-field array was sampled in two different ways, referred to here as the $\phi = 0°$ *cut*, and the *3-D sampled far-field pattern*. The $\phi = 0°$ *cut* contained 180 samples of the far-field radiation pattern, one for each θ integer interval in ($\phi = 0°, \theta \in [-90°, 90°)$). The *3-D sampled far-field pattern* contained 180 samples of the far-field radiation pattern in a (θ, ϕ) grid.

The 25-element regular dense array (a) was simulated for up to two simultaneous failures, $k_{dense} = 2$. From (4.1) in Section 4.2, this resulted in a total number of unique failure scenarios $R_{dense} = 326$. The 18-element irregular array (b) was simulated for up to three simultaneous failures $k_{irregular} = 3$, resulting in a total number of scenarios $R_{irregular} = 988$. The 8-element linear array (c) was simulated for all possible failure scenarios $k_{linear} = 8$, so that the total number of scenarios was $R_{linear} = 2^8 = 256$.

Support vector model kernels

The SVM described in this section was a one-vs-one SVM classifier [45]. The kernels that were compared were the radial basis function (RBF) kernel, the polynomial kernel of degree 1 (linear) kernel,

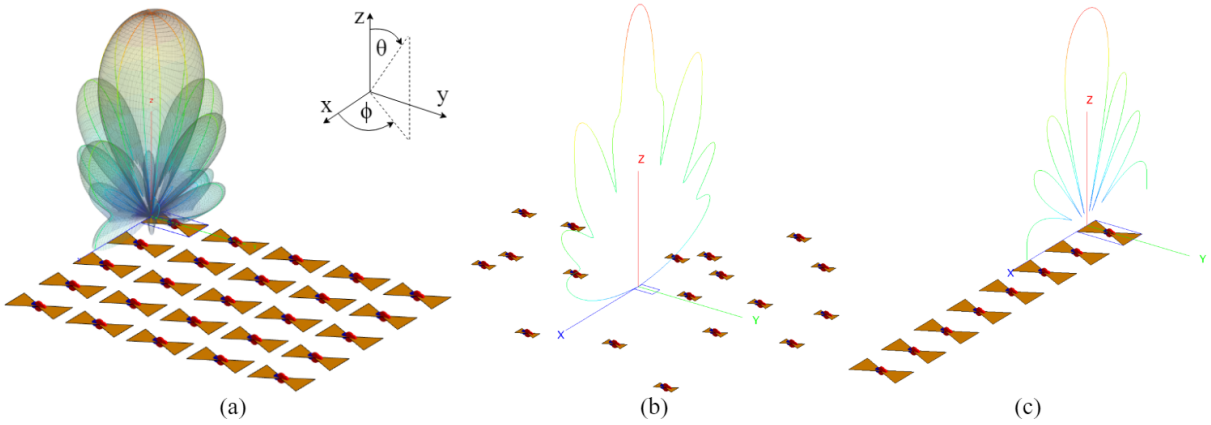


FIGURE 4.7: The (a) 25-element regular dense, (b) 18-element irregular (randomly spaced), and (c) 8-element linear antenna arrays used to generate the far-field data sets for this investigation.

and the polynomial kernel of degree 2. The classification accuracy of an SVM with the three different low-degree polynomial kernels on test data with increasing SNRs is plotted on a logarithmic scale in Fig. 4.8 to show up to what noise level the kernels under investigation were able to identify failed elements.

Fig. 4.8 shows the effects of the three kernels on the *3-D sampled far-field pattern* for all simulated failure scenarios. The two polynomial kernels consistently identified the correct patterns better than the RBF kernel, in accordance with the findings by Yeo [30]. The polynomial kernel of degree 1 (linear kernel) is used throughout the rest of the section to compare data sets further.

Sampling methods

The two sampling methods, the $\phi = 0°$ *cut*, and the *3-D sampled far-field pattern*, were compared for each configuration in Fig. 4.9.

The results of the linear and irregular array configurations in Fig. 4.9 show that the *3-D sampled far-field pattern* sampling method was slightly more reliable than the $\phi = 0°$ *cut* sampling method. This means that the *3-D sampled far-field pattern* sampling method contained more information that the SVM could use to distinguish between different scenarios. However, the difference in the reliability and final accuracy of the two sampling methods was small. This means that for a linear or irregular array, the *3-D sampled far-field pattern* sampling method did not contain much more information than the $\phi = 0°$ *cut* sampling method.

Each ideal training example of 180 features was a point in a feature space of 180 dimensions. Thus, for each configuration, R points were plotted to the 180-dimensional feature space. The one-vs-one

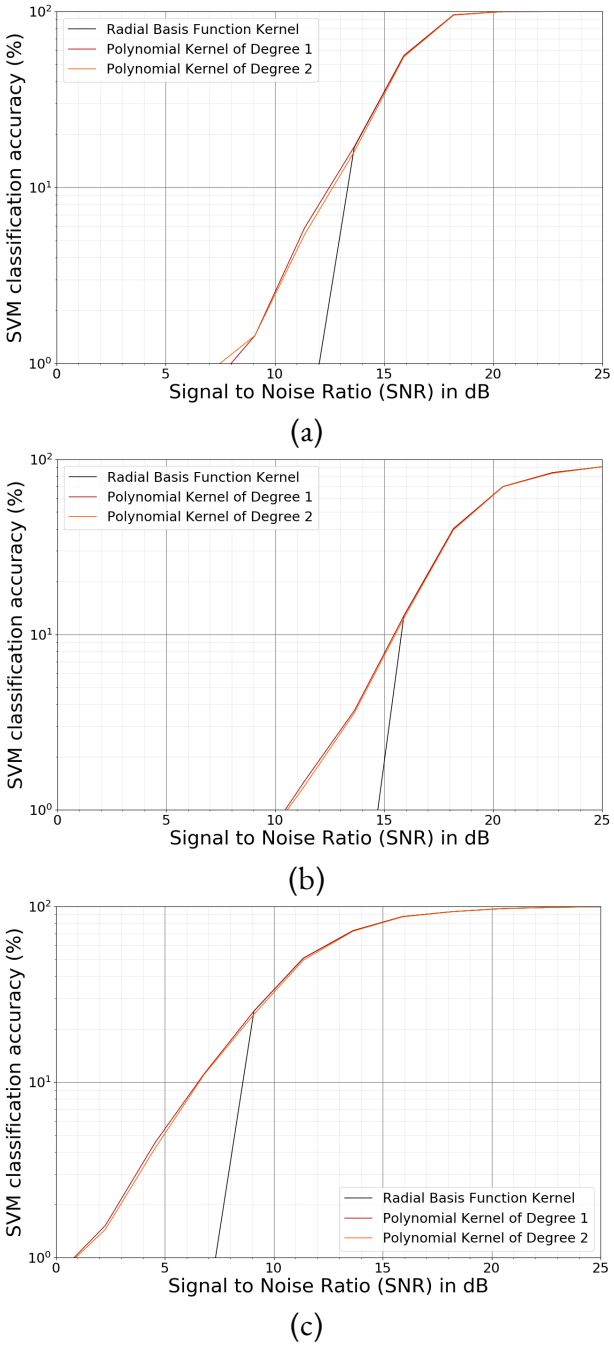


FIGURE 4.8: Support vector model kernel results for configurations (a) (25-element regular dense array, 326 classes), (b) (18-element irregular array, 988 classes) and (c) (8-element linear array, 256 classes).

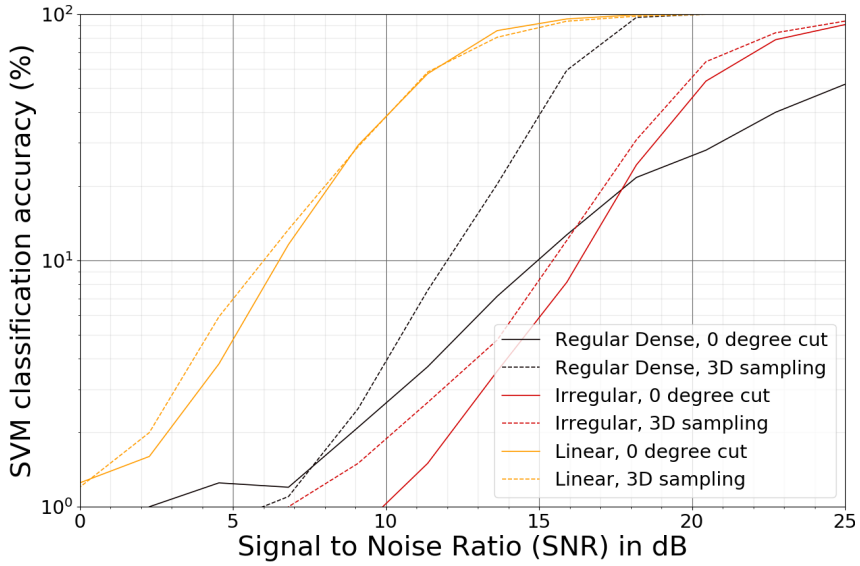


FIGURE 4.9: Comparing what the sampling method does to the SVM accuracy for a linear kernel, for each configuration.

SVM calculated a support vector that separated each pair of points in the feature space. As the number of unique failure scenarios (R) increased, the points plotted in the 180-dimensional feature space became more dense. This made it harder for the SVM to distinguish between classes, which resulted in the large difference in reliability between the linear ($R_{linear} = 256$) and irregular ($R_{irregular} = 988$) array configurations.

The configuration of a regular dense array is symmetrical around the $\phi = 0°$ plane. This meant that two different scenarios that were mirror images of each other across the $\phi = 0°$ cut, had exactly the same effect on the far-field amplitude sampled at $\phi = 0°$. This produced two identical patterns for two different classes in the data set, which made it impossible for the SVM to achieve high accuracies using the $\phi = 0°$ cut sampling method.

The case of the regular dense antenna array illustrates that the *3-D sampled far-field pattern* sampling method contains more information than the $\phi = 0°$ cut sampling method for antenna array configurations that are symmetrical around the $\phi = 0°$ plane.

Number of classes

Fig. 4.8 shows that, as the number of classes in the data sets increased, the SNR at which the SVM was able to reliably identify the true pattern from noisy signals was higher. To see whether this trend was because of the number of classes the SVM had to discern, only the first 200 classes of each data set were used to train and test the SVM. The results of using the full number of classes and only 200 classes to train the SVM, for each configuration are compared in Fig. 4.10.

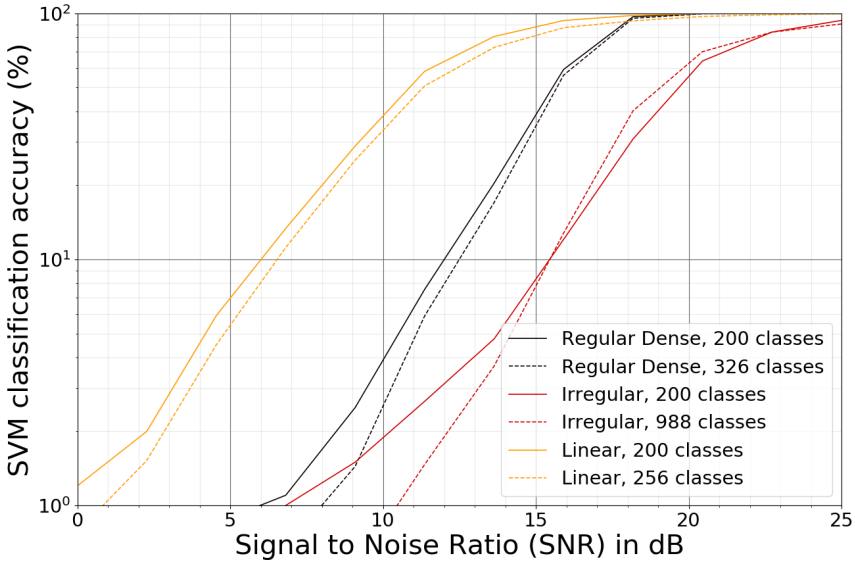


FIGURE 4.10: Comparing what the number of classes does to the SVM accuracy for a linear kernel, for each configuration. The sampling method was kept as 3-D sampling for all configurations.

The results of the linear array configuration in Fig. 4.10 illustrate that, when the number of classes in a data set was decreased, the points in the 180-dimensional feature space were fewer and further apart. This made it easier for the SVM to distinguish between classes. As a result, the reliability and accuracy of the SVM improved slightly. However, there was still a large jump in reliability between data set configurations. This means that the array configuration and the number of elements in the array affected the distribution of the points in the 180-dimensional feature space.

As the number of elements in a regular (symmetrical) array configuration increases, the effect of the failure of one element on the shape of the far-field pattern becomes less pronounced. The points that represented these far-field patterns in the 180-dimensional feature space were situated closer together as the number of elements increased. Thus, for symmetrical configurations, as the number of elements in the array increases, the SVM is less and less reliably able to distinguish between classes. However, the failure of an element in a symmetrical array caused the side-lobes of the array's far-field pattern to rise, which made the effect of a failed element in a symmetrical array more predictable, as the far-field pattern should also be symmetrical.

Irregular arrays were less affected by element failure. They were designed so that the effects of individual elements on the side lobes of the far-field pattern cancelled each other out, resulting in low, asymmetrical side lobes. When one element failed, the functional elements still kept the far-field side lobes low. Consequently, for irregular array configurations, the failure of one element had a much smaller effect on the shape of the far-field pattern than for a symmetrical configuration with the same

number of elements. In the representation of the far-fields of different failure scenarios as points in the 180-dimensional feature space, the classes for an irregular array lie much closer together than in a regular array with the same number of elements. This means that, as the number of elements in an irregular array increases, the effect of the failure of one element on the shape of the far-field pattern becomes less predictable, and it is harder for the SVM to distinguish between classes.

4.3.3 Conclusion

The methods introduced by Yeo [30] to categorise failure scenarios with a binary code, and compare the ability of different SVM kernels to classify far-field patterns with worsening SNRs were implemented and tested in three different simulated scenarios. Two sampling methods were compared, the $\phi = 0°$ *cut* and the *3-D sampled far-field pattern*.

This work shows that it is important to bear in mind the configuration of the antenna array when choosing how to sample the far-field pattern to generate training data. Consistent with Yeo's findings [30], Fig. 4.8 shows that an SVM with a linear kernel was functional for data sets generated by arbitrary array configurations. However, Fig. 4.9 illustrates that, for symmetrical array configurations, it was better to sample the far-field pattern to be used as training data with the *3-D sampled far-field pattern* sampling method. Symmetrical configurations sampled using the $\phi = 0°$ cut sampling method produce classes that look identical and result in poor SVM accuracy.

Finally, as the number of elements in an arbitrary array increases, the effect of one element on the amplitude of the far-field pattern becomes less noticeable. This means that, for the application of large antenna arrays with thousands of elements, it could become impossible for the current SVM method to locate one failed element.

4.4 Conclusion

This chapter describes how failed antenna elements were detected and located using various classification algorithms. The two algorithms investigated in more depth were the FNN and SVM. The algorithms were trained on sampled far-field patterns.

4.4.1 Sampling methods on a feedforward neural network

The first experiment investigated far-field sampling methods. In the first part of the experiment, an FNN was trained with multi-label data, so that each example had the same number of labels as the number of antennas. That means for an antenna array of 25 elements, such as the array shown in Fig. 4.1, each example had 25 labels. Each label indicated the state (0=ON, 1=OFF) of one antenna.

Ten different sampling methods, described in Table 4.2, were used to sample the 3-D far-field pattern of each unique failure scenario, to generate 10 sampling method training data sets. The same FNN model was trained on each sampling method data set, and the results were compared according to accuracy and training time:

- The *diagonal cuts* sampling method achieved the highest accuracy (90.91% accuracy).
- The *3-D pattern (182 samples)* method achieved the shortest training time.

The *3-D pattern (182 samples)* method is recommended for training sets with more scenarios, e.g. a larger n or k in (4.1), where the difference in training time between sampling methods is likely to become even more significant.

4.4.2 Sampling methods on classification algorithms

In the second part of the first experiment, 14 other out-of-the box scikit-learn multi-label classification algorithms were trained on the ten sampling method data sets and compared for their accuracy. Most algorithms did not fare well but the four best algorithms were the FNN, the one-vs-rest classifier with a linear support vector classifier, the one-vs-rest classifier with logistic regression, and the one-vs-rest classifier with logistic regression and a cross-validation core. The one-vs-rest classifier with logistic regression and a cross validation core was the most successful overall, and achieved 100% accuracy. From this part of the experiment, we can see that not all classification algorithms have the same behaviour as an FNN, where the number of parameters has a strong influence on the accuracy of the algorithm. From the results in Fig. 4.9, it was clear that the 3-D sampling methods consistently contain more distinguishing information than the single cut-composed sampling methods.

4.4.3 Effect of array configuration on model performance

The second experiment investigated array configurations. Three different array configurations, as shown in Fig. 4.7, were used to train an SVM with different kernels on test data with worsening SNRs, a method first described by Yeo [30]. In the first part of the experiment, all configurations were compared on three kernels according to their accuracies on training samples with worsening SNRs. The linear kernel yielded the best performance on noisy signals, which was consistent for all configurations. This result compares well with what was found by Yeo [30]. The result was also in line with what was found in Table 4.4 which is that the most linear and basic classification algorithms performed best on the far-field data. This may be because the training data has a near-linear line-of-separation, which is difficult to model with high-polynomial kernels or complex machine-learning models.

This experiment demonstrated the importance of being aware of the array configuration type. Taking a single cut through the origin (such as the $\phi = 0°$ cut or the $\phi = 90°$ cut) of symmetrical configurations, such as a regular dense or linear antenna array, will result in some classes having identical patterns wherever the OFF antennas are a mirror image of each other. This makes it impossible for any classification algorithm to achieve high accuracy. This agrees with what was found in the first experiment, that it was better to use the 3-D sampling method, if the number of samples are to be kept low.

Consistent with what was found by Yeo [30], Fig. 4.8 shows that an SVM with a linear kernel is functional for datasets generated by arbitrary array configurations. However, Fig. 4.9 illustrates that, for symmetrical array configurations, the *3-D sampled far-field pattern* sampling method is better when sampling the far-field pattern to be used as training data. When symmetrical configurations are sampled using the $\phi = 0°$ cut sampling method, the resultant classes look identical, which produces low SVM accuracy.

Finally, as the number of elements in an arbitrary array increases, the effect of one element on the amplitude of the far-field pattern becomes less noticeable. This means that, for the application of large antenna arrays with thousands of elements, it could become impossible for the current SVM method to locate a singular failed element.

5 Case Study: Failure Detection in a Manufactured 16-element Circular Patch Antenna Array based on Far-field Sampling

5.1 Introduction and Aim

The previous chapter described how FNNs and SVMs were applied to simulated data to establish whether it was theoretically possible to detect failed elements in an arbitrary antenna array using the simulated far-field pattern as training data, as proposed by many researchers in the literature reviewed. Then we needed to establish whether machine-learning algorithms such as those presented in Chapter 4 would be useful when the algorithms were trained on simulated far-field patterns, but tested on measured far-field patterns. The next step for the current study was therefore a practical test of our proposed methods on a measured example. According to the literature reviewed, this had seldom been done before [39].

In this chapter, we describe how we designed and manufactured a 16-element circular patch antenna array so that we could compare the principle and diagonal cuts of its simulated and measured far-field radiation patterns.

The aim was to investigate whether it was feasible to train a machine-learning algorithm, such as an FNN or SVM, on a numerically efficient simulated far-field data set, and use the trained model to identify the failure scenario when given a measured far-field pattern.

The hypothesis was that the shape of the far-field cuts, even for such a numerically efficient simulation, would contain enough information for a machine-learning algorithm such as an SVM or FNN to recognise the same patterns in the far-field measurements of the same manufactured array.

The objectives and setup of this case study are summarised in Table 5.1.

The experiment limitations, design and setup are described in Section 5.2. The design of the antenna array is explained in Section 5.3. The steps followed to manufacture the antenna array design are outlined in Section 5.4. The measurements necessary for the six failure scenarios are expounded

TABLE 5.1: The setup and objectives of the Measurements Case Study described in Chapter 5.

	Measurements Case Study
Description	Investigation into whether it is feasible to use the simulated far-field pattern as training data for a machine-learning algorithm designed to identify failures in a measured far-field pattern.
Training Data Sets	The $\phi = 0°$, $\phi = 90°$, $\phi = 45°$ and $\phi = 135°$ cuts of 6 unique failure scenarios (see Table 5.3) were simulated and compared to the measured cuts.
Array Configurations	16-element (4×4) circular patch antenna array with up to 3 simultaneous failures (see Fig. 5.2).
Objectives	To simulate and measure the far-field pattern of the same antenna array. To establish whether the patterns were similar enough for a machine-learning algorithm trained on *simulated* far-field radiation patterns to identify failed elements when given a *measured* far-field radiation pattern.

in Section 5.5. The simulated and measured far-field patterns are compared in Section 5.6. The conclusion that the raw simulated data should not be used as-is as training data for a machine-learning algorithm whose purpose is to identify failed elements from a measured far-field pattern is expounded in Section 5.7.

5.2 Methodology and Experimental Setup

This section explains the limitations on the experiment and how they defined the design choices. The calculations for the patch diameter and other parameters of the circular patch are summarised in Section 5.3 and listed in Table 5.2. We describe the various scenarios used to compare the simulated and measured far-field patterns in the experimental setup in Section 5.4.2.

5.2.1 Design choices

The requirements for this case study were:

- that the antenna array was easy to design and simulate in FEKO;
- that it was possible and simple to manufacture the antenna array at Stellenbosch University;
- that the antenna array design fit the frequency and size restrictions of the spherical near-field scanner at Stellenbosch University;
- that the antenna array could be powered by a power combiner available at Stellenbosch University;

- that it would be easy to attach and remove cables from the antenna array, to emulate failed elements; and
- that the simulation of the failure scenarios was computationally inexpensive.

In order to meet the requirements listed above, the following design choices were made:

- A circular patch antenna is a microstrip type of antenna, easy to simulate in FEKO and to manufacture using the PCB milling machine available at Stellenbosch University. Rogers RO4350B PCB plates [79] were readily available at Stellenbosch University. The standard panel size available is 305×457 mm. The antenna elements were therefore designed to fit this plate size.
- A 16-way radial power combiner that had been designed at Stellenbosch University [80] split the power to the antennas equally, without allowing reflections from one antenna to affect another in the array. Thus, when an antenna failed, the other antennas in the system were not affected. The power combiner had 16 ports and worked at frequencies around 4 GHz. Using it to power the antenna array implies that the antenna array should have 16 elements and a centre frequency near 4 GHz.
- To easily emulate antenna failures while taking measurements, the cable of a "failed" antenna was simply detached from both the antenna and the power combiner.
- To keep the simulation of the failure scenarios computationally inexpensive, an infinite ground plane was used to simulate each unique failure scenario. The simulations had to be computationally economical because, as the number of elements and the number of simultaneous failures increases, the time it takes to simulate all possible failure scenarios increases proportionally.
- Finally, the coordinate system of the spherical near-field scanner used to take the measurements was different from the spherical coordinate system used in FEKO. To easily compare the two far-field patterns, their principal and diagonal cuts were compared, instead of using 3-D patterns.

The spherical near-field scanner at Stellenbosch University, developed by NSI-MI Technologies is NSI-MI's most common, "Roll over Azimuth" spherical system. The scanner can measure antennas that radiate at frequencies between 750 MHz and 26.5 GHz [81], and that have a diameter between 0.25 m and 3 m.

A 16-element circular patch antenna array that radiates at a centre frequency f_r of 3.87 GHz was chosen because it met all the design requirements. The design of the circular patch is discussed in Section 5.3.

5.3 Designing the Antenna Array

In [50], Balanis arrived at the following formulae to determine the radius (a) of a circular patch antenna when the desired resonant frequency (f_r), the dielectric constant of the substrate (ϵ_r), and the height of the substrate (h in cm) are known.

$$a = \frac{F}{\sqrt{1 + \frac{2h}{\pi\epsilon_r F}[\ln(\frac{\pi F}{2h}) + 1.7726]}}, \tag{5.1}$$

where

$$F = \frac{8.791 \times 10^9}{f_r\sqrt{\epsilon_r}}. \tag{5.2}$$

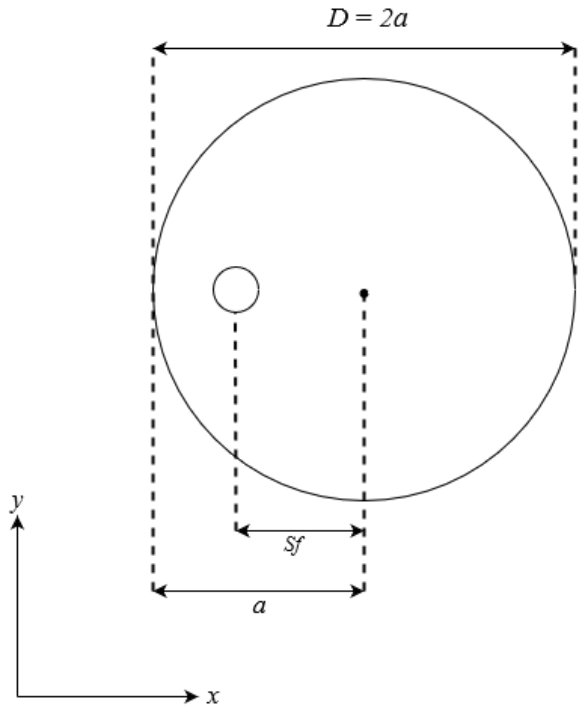
From the parameter values listed in Table 5.2, using the Rogers substrate and a power combiner that functions around 4 GHz, the diameter $D = 2a$ of the circular patch antenna was calculated to be 22.768 mm. After optimising the design using FEKO [48] and Antenna Magus [55], the design parameters were finalised as listed in Table 5.2. The final design resulted in an antenna array that fit on a 200 mm $\times$ 200 mm PCB. The spacing between antennas is $\frac{\lambda}{2}$. The parameters that describe the geometry of the circular patch in Table 5.2 are defined in Fig. 5.1. The diagram is for illustrative purposes, and is not drawn to scale.

TABLE 5.2: Parameters of the circular patch antenna

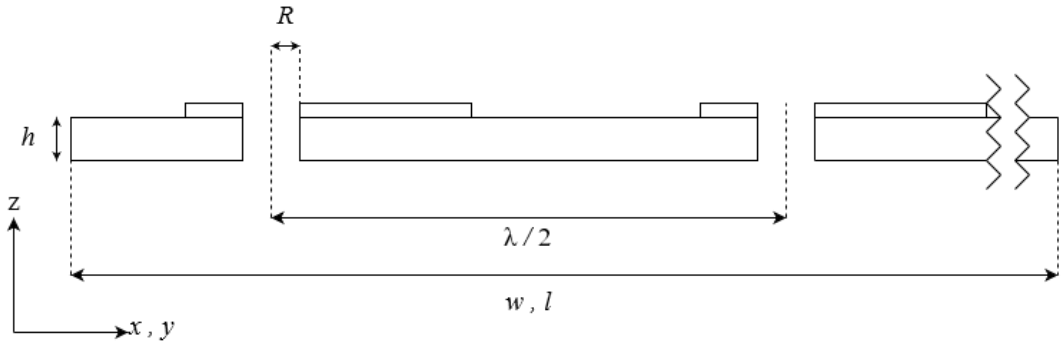
Parameters		
Symbol	***Value***	***Description***
f_r	3.87 GHz	Desired resonant frequency
ϵ_r	3.58	Relative permittivity
h	813 μm	Substrate height
w, l	200 mm	Substrate width and length
R	190 μm	Feed pin radius
$\tan(\delta)$	2.4e-3	Loss tangent of the substrate medium
D^*	22.865 mm	Patch diameter
S_f^*	2.49 mm	Feed offset

* Values deduced from calculation and refined using simulation software [48], [55].

The steps followed to manufacture the antenna array design are outlined in Section 5.4.



(A) Top view of a circular patch



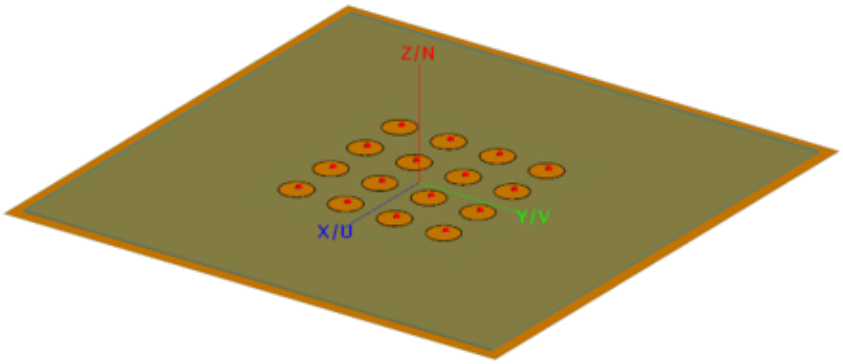
(B) Side view of the antenna array design

FIGURE 5.1: The parameters that describe the geometry of the circular patch as listed in Table 5.2 are shown using a top and side view.

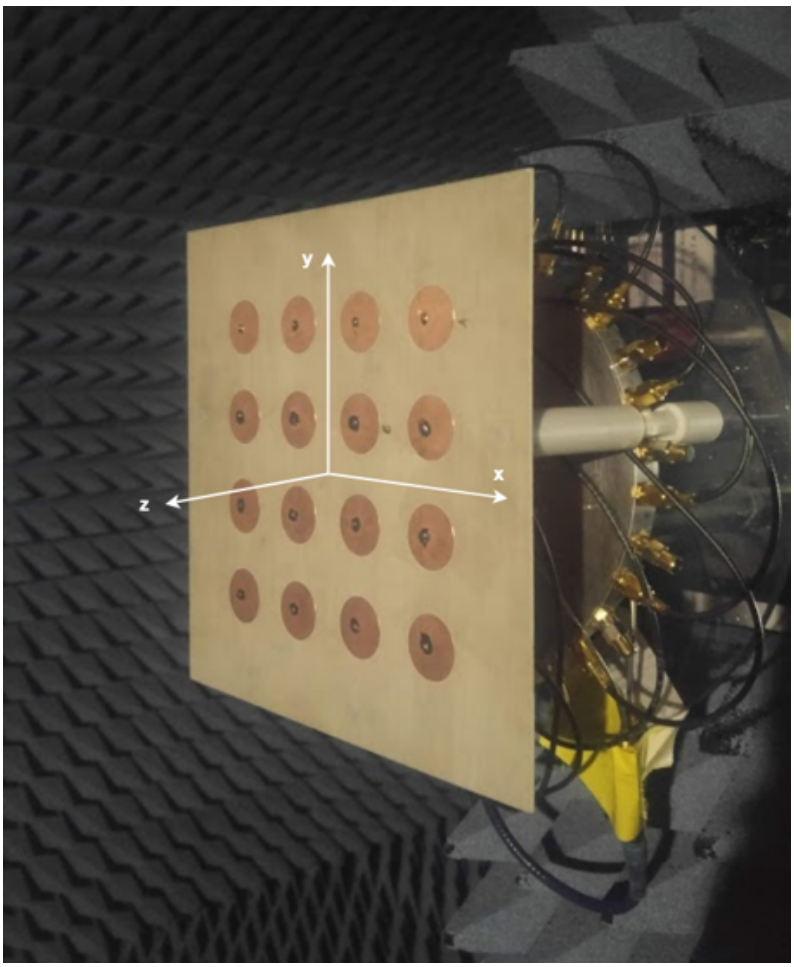
5.4 Manufacturing the Antenna Array

This section describes how the antenna array was put together so that element failures could be easily simulated. It justifies the comparison of the measured and simulated reflection coefficients to show whether the manufactured array performed as expected.

The FEKO model (see Fig. 5.2 (A)) containing the array dimensions was used to create the file given to the PCB milling machine to cut the shapes of the 16 antennas onto a Rogers PCB plate. Three



(A) Simulated array



(B) Manufactured array

FIGURE 5.2: In (A), the 16-element circular patch antenna array is shown as it was designed and simulated in FEKO. In (B), the 16-element circular patch antenna array is shown mounted on the spherical near-field scanner in the anechoic chamber at Stellenbosch University.

support structures were designed and 3-D printed to hold the antenna array and the power combiner in place when they were rotating in the spherical near-field scanner. Female SMA connectors were soldered onto the antenna elements so that the antennas could be pin-fed through the ground plane. Sixteen cables were cut at exactly the same length, and male SMA connectors were attached to each end to connect each antenna to the power combiner. The assembled antenna array can be seen mounted on the spherical near-field scanner in Fig. 5.2.

5.4.1 Reflection coefficients

As mentioned in Chapter 2, the reflection coefficients can be used to determine the bandwidth and centre frequency of an antenna. A comparison of the measured and simulated reflection coefficients can show whether the manufactured array performs as it is expected to.

The reflection coefficients of antennas A and B were measured and compared to the simulated reflection coefficients as shown in Fig. 5.3. From the figure, it is clear that the manufactured antenna array resonates at $f_r = 3.78$ GHz, as expected from the simulation. Before the simulated and measured far-field patterns were compared, the parameters of the simulated model were manually adjusted until the simulated and measured reflection coefficients matched as well as possible. Of the parameters listed in Table 5.2, the quantity that impacted the model the most was ϵ_r.

5.4.2 Experimental setup for three simultaneous failures

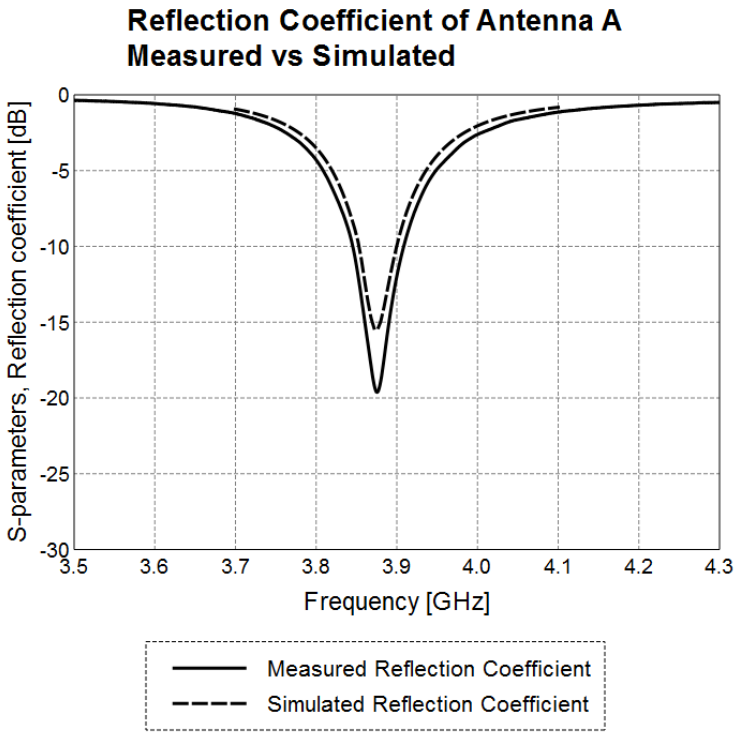
The layout of the 16-element circular patch antenna array is shown in Fig. 5.4. The measured and simulated far-field cuts were compared for up to three simultaneous failures. The figure indicates the placements of the three antenna elements A, B and C, that were turned OFF or ON in different combinations.

The measurements necessary for the six failure scenarios are expounded in Section 5.5.

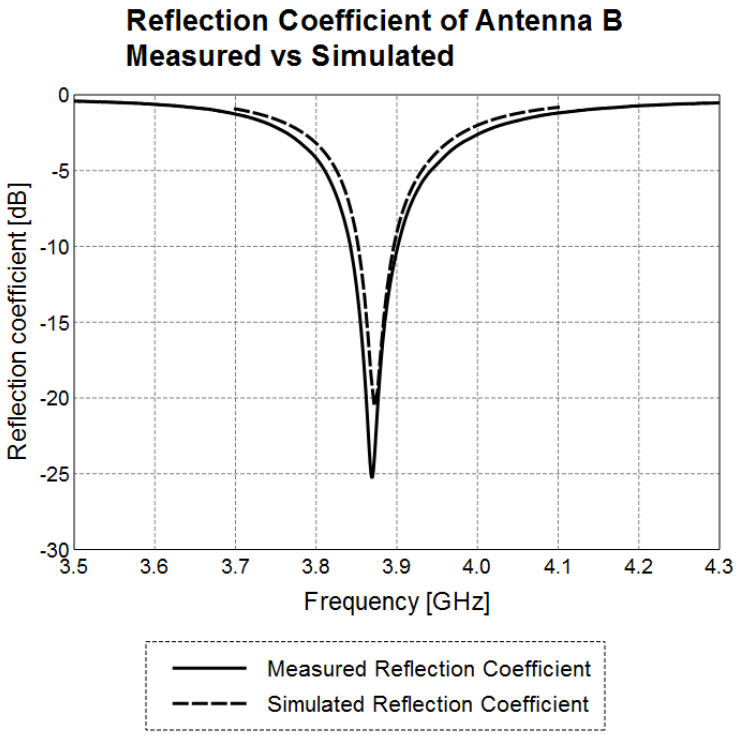
5.5 Measuring the Failure Scenarios

The antenna array was measured for the six different failure scenarios described in Table 5.3. The table uses the method introduced for Table 4.5 in Section 4.3 to present the state of all elements per unique failure scenario as a binary code. The SID is the number that identifies the simulated failure scenario.

The measurement identification number (MID) is the number used to identify the measured scenario in this text. Of the six unique failure scenarios were emulated and measured, the first MID was where all the antennas were fully functional. For the second MID, only antenna A was turned OFF. For the third MID, antenna B was turned OFF. For the fourth MID, antennas A and C were turned



(A) Antenna A



(B) Antenna B

FIGURE 5.3: A comparison of the simulated and measured reflection coefficients of antenna A and antenna B.

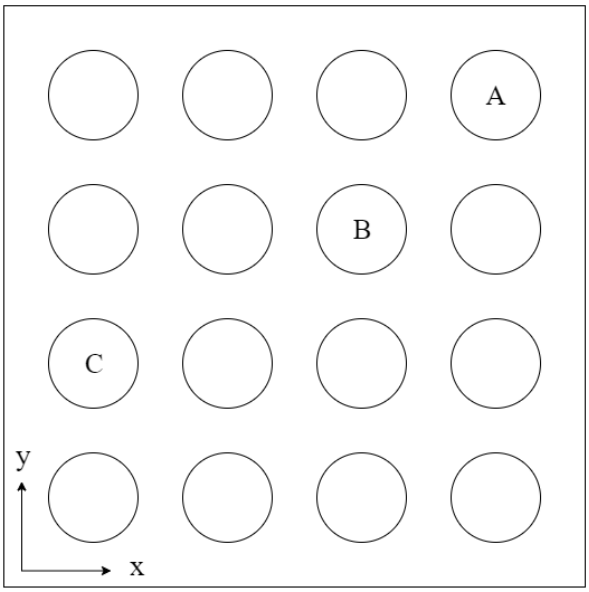


FIGURE 5.4: Layout of 16-element circular patch antenna array. The indicated elements A, B and C were switched off in different combinations to create six failure scenarios.

off. For the fifth MID, antennas A and B were turned OFF. Finally, for the sixth MID, antennas A, B and C were turned off. Table 4.5 shows the SID to which each MID corresponded.

TABLE 5.3: Measured scenarios

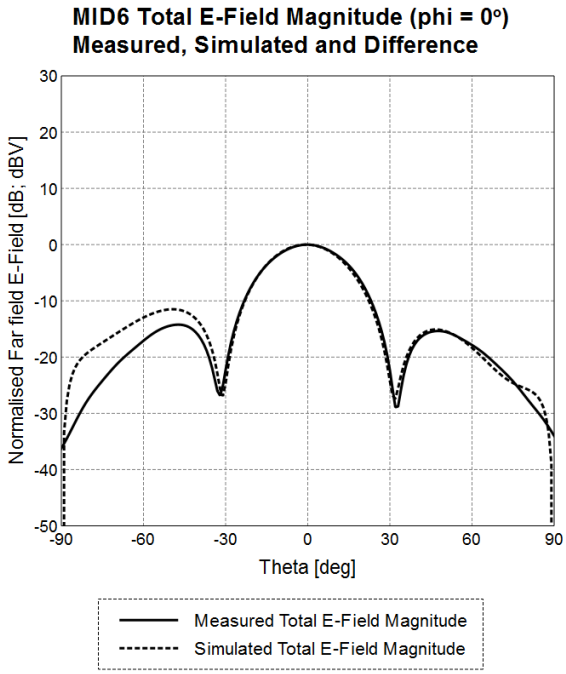
MID[a]	SID[b]	Binary encoding	Description
1	1	0000000000000000	All elements ON
2	14	0000000000001000	Element A OFF
3	11	0000000001000000	Element B OFF
4	106	0000000100001000	Elements A and C OFF
5	119	0000000001001000	Elements A and B OFF
6	623	0000000101001000	Elements A, B and C OFF

[a] MID: measurement identification number
[b] SID: scenario identification number

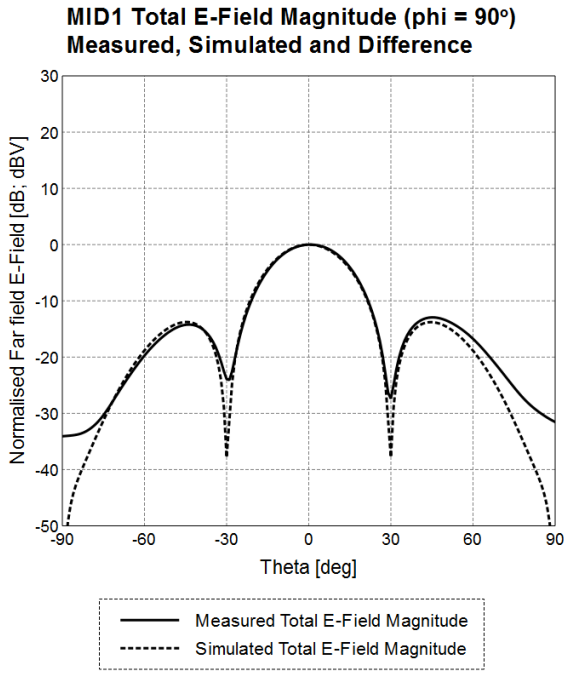
The simulated and measured far-field patterns are compared in Section 5.6.

5.6 Comparing Far-field Patterns

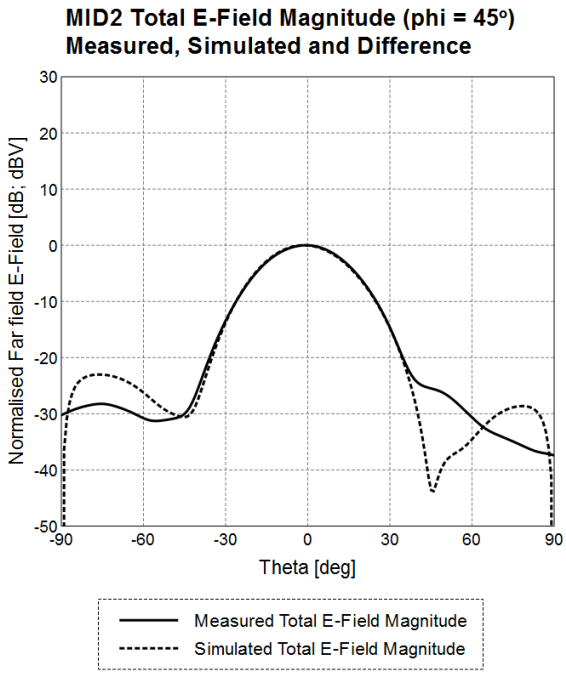
This section describes how we compared the simulated and measured far-field patterns of the antenna array designed in Section 5.2. We show figures of the measured and simulated patterns and discuss



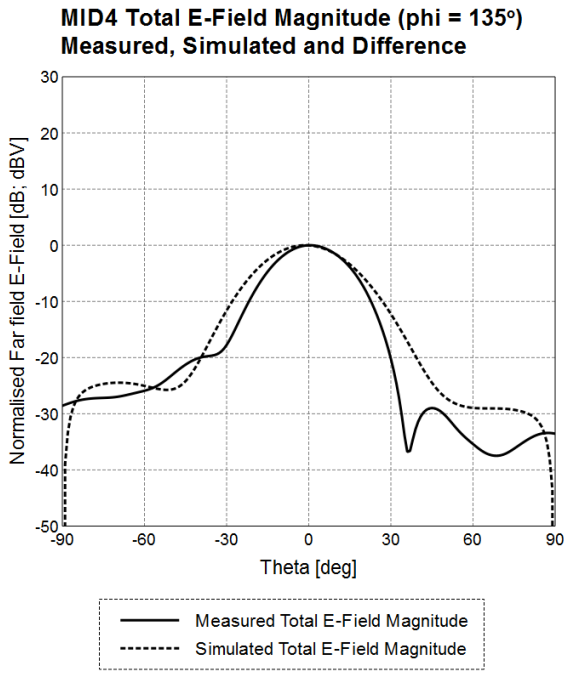
(A) 0° cut, MID6



(B) 90° cut, MID1



(C) 45° cut, MID2



(D) 135° cut, MID4

FIGURE 5.5: Four patterns of the principal and diagonal cuts of the simulated and measured far-field patterns of each MID. The comprehensive set can be found in Appendix A.

why it is not feasible to use the shape of the simulated far-field patterns as-is as training data for a machine-learning algorithm designed to identify failed elements in a measured far-field pattern.

Once the design for the antenna array was finalised, we simulated the far-field patterns of all possible failure scenarios for up to three simultaneous failures, resulting in 697 unique failure scenarios.

A spherical near-field scanner in an anechoic chamber was used to measure the far-field patterns of each of the six MIDs in Table 5.3. The anechoic chamber isolated the antenna under test from outside interference sources. To emulate an antenna failure, the relevant antenna's cable was detached from the antenna and the power combiner, and the port on the power combiner was terminated with a matched (50-Ω) load.

The principal and diagonal cuts of the simulated and measured far-field patterns of each MID listed in Table 5.3 are shown in Appendix A. Four of these patterns are shown in Fig. 5.5 to demonstrate our findings.

The simulated pattern is the ideal representation of the far-field pattern. Consequently, the dB value strives to $-\infty$ at $\theta = -90°$ and $\theta = 90°$. The spherical near-field scanner is not capable of measuring such low dB values, so the measured far-field pattern rarely registers a value below -40 dB.

On average, there was a smaller difference between the principal cuts than between the diagonal cuts. The positions of the first nulls are modelled better in the principal cuts than in the diagonal cuts.

The results show that the simulated pattern usually has deeper nulls than the measured pattern. The average difference between the simulated and measured results for the main lobe is small. However, the results differ severely between the simulated and measured patterns in the side lobes.

5.7 Conclusion

To make it possible for a machine-learning algorithm to classify a given test case, it is important for the training data to resemble as closely as possible the data that has to be classified in the testing phase. In this case study, we chose to investigate cuts in the ϕ direction of the amplitude of the far-field pattern as the training and testing data.

5.7.1 Amplitudes affected by nonidealities

From Fig. 5.5, and the comprehensive set in Appendix A, it is clear that the inexpensive simulation of the 16-element circular patch antenna array did not model the measured far-field pattern accurately enough. The difference between the measured and simulated far-field patterns illustrates how the amplitude of the measured far-field pattern was affected by nonidealities that were not modelled by the numerically efficient simulation (for example, small deviations in the simulated parameters listed

in Table 5.2), even though these measurements were taken in an anechoic chamber that isolated the antenna under test from sources outside the anechoic chamber. For larger arrays, such as those envisioned for the SKA radio astronomy project, an accurate measurement of the far-field poses even more challenges.

5.7.2 Added noise is irrelevant

In previous work, researchers added noise to the simulated far-field pattern to increase the training data set and make the machine-learning model robust enough to handle small deviations from the ideal pattern [30]. However, the measured far-field pattern in this study was not noisy, and nor was the measured far-field pattern in [39]. Noisy signals should therefore not be used as training data for a machine-learning algorithm if the purpose is to identify the failed elements in a measured signal.

5.7.3 Multiple simulations per failure scenario

Future work should rather simulate an array configuration many times per failure scenario: Each time the scenario is simulated, small random deviations can be made in parameters that have an effect on the shape of the pattern (for example, antenna geometry and material properties, such as those listed in Table 5.2).

6 Conclusion

6.1 Review

Machine learning is a new and fast-evolving field, which makes it difficult for other disciplines to keep abreast of the possibilities and, more important, the limitations of machine learning in their own contexts. Machine-learning practitioners, on the other hand, typically need to develop quite detailed insight into the peculiarities of the other discipline before they can successfully ply their trade.

This study was an example of such an interdisciplinary investigation. It required significant applied insights from both machine learning and antenna theory and practice in order to determine the feasibility of using machine learning to manage the system health of large antenna arrays like the SKA.

This thesis aimed to identify a trustworthy means of early detection and isolation of faulty elements in the SKA radio astronomy project [16] to improve the reliability of measured data. Once failed elements are identified, antenna failure correction methods [8]–[11] can be used to exclude failed elements from calculations.

NASA has used machine learning in the context of radio astronomy to detect element failures in the DSN, which is an international network of antennas [37]. Machine learning for element failure detection has also been investigated by researchers in military and satellite applications of antenna arrays. The trend is to use a cut of the amplitude of the far-field radiation pattern as training data. Machine learning algorithms such as genetic algorithms [31], [33], [39], [43], artificial neural networks [38], [44], case-based reasoning [31], [32], support vector models [30], [40], the Woodward-Lawson method [32], and bacteria foraging optimisation [41] have been investigated.

6.1.1 Effect of input data on accuracy

Previous work [33], [38], [39], [41], [44] has proven that it is theoretically possible to identify failed elements from the far-field pattern. However, not many researchers have looked at the effect of the input data on the accuracy of the machine-learning algorithm. In Chapter 4, we described how we therefore investigated:

- the effect of 10 far-field sampling methods on an FNN's accuracy and training time;

- the effect of 10 far-field sampling methods on the accuracies of different out-of-the-box classification algorithms; and
- the effect of the configuration of the antenna array on the accuracy of an SVM.

A detailed summary of the experiments' setup and objectives is given in Table 4.1.

Two articles were written from work presented in this thesis for international peer-reviewed IEEE conferences [46], [47].

6.1.2 Comparison of measured and simulated far-field patterns

As shown in Chapter 4, it has been illustrated that the far-field pattern can theoretically be used as training data to detect failed elements. Until now, very few researchers have validated their proposed techniques on a manufactured array [39]. Chapter 5 described how we measured and simulated a 16-element circular patch antenna array. The aim was to determine whether it is feasible to use the *simulated* far-field pattern as training data for a machine-learning algorithm designed to identify failures in a *measured* far-field pattern.

6.2 Experiment and Case Study Conclusions

In Chapters 4 and 5, we found that various parameters have an effect on the accuracy of the machine-learning algorithm. This includes the sampling method, algorithm choice, array configuration and the difference between the measured and simulated radiation pattern. The number of elements in the array and whether the array is symmetrical or asymmetrical also has an effect on the accuracy of the machine-learning model.

6.2.1 Effect of sampling method

In Chapter 4, we described how we found that training an FNN on a dataset using the *diagonal cuts* sampling method achieved the highest accuracy (90.91%). The *3-D pattern (182 samples)* method achieved a shorter training time, and a similar accuracy (87.88%). We therefore recommend the *3-D pattern (182 samples)* method for training data sets with a large number of elements (such as those planned for the SKA). The difference in training time is likely to become more significant as the number of elements and the number of simultaneous failures in the array increases (from (4.1) in Section 4.2).

Fourteen out-of-the-box scikit-learn multi-label classification algorithms were also trained on the ten sampling method data sets and compared for their accuracy. Most algorithms did not fare well,

but the one-vs-rest classifier with logistic regression and a cross-validation core was the most successful overall, with a 78.18% average accuracy over all data sets. From the results in Fig. 4.5, it is clear that, generalised over different classification algorithms, the 3-D sampling methods contain more distinguishing information than single-cut composed sampling methods.

6.2.2 Choice of SVM kernel

The three different array configurations shown in Fig. 4.7 were used to train an SVM with different kernels on test data with worsening SNRs. The linear kernel yielded the best performance for all configurations (consistent with what was found by [30]), and was used throughout the rest of the experiment. The findings in Table 4.4 and the finding that the linear kernel performed best, suggest that the training data over all configurations has a near-linear line-of-separation, which is difficult to model with high-polynomial kernels or complex machine-learning models.

6.2.3 Effect of symmetry

This experiment demonstrated that it is important to consider the symmetry of the array configuration. In symmetrical configurations such as a linear or regular dense array, taking a single cut through the origin (such as the $\phi = 0°$ cut or the $\phi = 90°$ cut), will result in some classes having identical patterns wherever the OFF antennas are a mirror image of each other. This makes it impossible for any classification algorithm to achieve high accuracy. This agrees with what was found in the first experiment, that it was better to use the 3-D sampling method, if the number of samples must be kept low. This is illustrated in Fig. 4.9, where the *3-D sampled far-field pattern* sampling method, that samples the far-field pattern in a θ, ϕ grid, achieves higher accuracies on symmetrical configurations than using the $\phi = 0°$ *cut* sampling method, which only takes a single cut of the far-field pattern in one plane.

6.2.4 Effect of asymmetry

As well as testing the effects of regular dense configurations, we tested the effects of an irregular sparse configuration on SVM accuracy. We found that the effect of one antenna on the far-field radiation pattern becomes less noticeable as the number of antennas in the antenna array increases. If the antenna array has an irregular sparse configuration, the effect of one antenna on the radiation pattern is even less pronounced. This means that, for application in large irregular arrays with thousands of elements, it could become impossible for the current SVM method to locate a single failed element.

6.2.5 Effect of pattern source

In Chapter 5, the difference between each MID and the equivalent SID was calculated for each MID. We demonstrated that the simulated pattern, although it closely resembled the far-field pattern generated by the array, would not work well as training data for a machine-learning model. This is because the measured pattern and the simulated patterns have been generated by two different sources, which means that the far-field radiation pattern, and therefore training data is inherently different. A machine-learning algorithm should be able to identify the class of a new, unseen example after it has been trained on the training data. Therefore, the training data must resemble this test data as closely as possible.

Simulated data therefore cannot be used as-is as training data for a machine-learning algorithm that must identify failed scenarios from a measured far-field pattern.

The differences between the measured and simulated far-field patterns of a specific scenario might be overcome by calculating the derived quantities, such as the half-power beam-width, location of first nulls (but not the depth of the nulls), and the side-lobe levels.

6.3 Overall Conclusions

We found that it would not be feasible, at this point in the evolving history of technology, to employ machine learning to detect single antenna failures by measuring distortions in the far-field radiation patterns generated by a very large array of antennas in an irregular sparse configuration.

Chapter 4, Section 4.3, showed that the effect of one element on the far-field radiation pattern becomes less discernible as the number of elements increase. If the antenna array has an irregular sparse configuration, the effect of one antenna on the radiation pattern is even less discernible.

As was demonstrated in Fig. 1.2, SKA-mid, a part of the SKA radio astronomy project, is planned to be an irregular sparse antenna array of more than 200 elements. Even if it were possible to measure the far-field pattern, to train on simulated data and to test using the measured data, the failure of one antenna will have almost no effect on the total far-field radiation pattern of the antenna array. The different failure scenarios will be relatively close to each other in the multi-dimensional feature space, which would make it difficult for a machine-learning algorithm to distinguish between failure scenarios.

6.4 Recommendations for Future Work

In Chapter 5, we illustrated that the simulated far-field pattern does not reproduce the measured far-field pattern well enough for it to be used as training data. The difference between the simulated far-field pattern and a far-field pattern measured outside an anechoic chamber will be even greater than the difference between the simulated pattern and data measured inside an anechoic chamber. We therefore do not recommend using the amplitude of the simulated far-field pattern as-is for element failure detection in large antenna arrays. This is significant, as the most common technique described in literature currently available for training machine-learning algorithms, is to train them on the amplitude of the far-field pattern of an antenna array [32], [33], [38]–[41]. The far-field data should be preprocessed, for example by calculating the derived quantities (e.g., the half-power beamwidth, location of the first nulls and side lobe levels, as demonstrated in Fig. 2.5) before it is used as training data for a machine-learning algorithm.

If the far-field pattern is chosen as the training data, adding noise to the training data to increase the training data set will not model the measured far-field pattern well. Future work should rather focus on simulating an array configuration many times per failure scenario and introducing small random deviations in parameters that may have an effect on the shape of the pattern. Parameters such as antenna geometry and material properties can be used. Non-idealities can also be introduced in the simulation, such as interference sources.

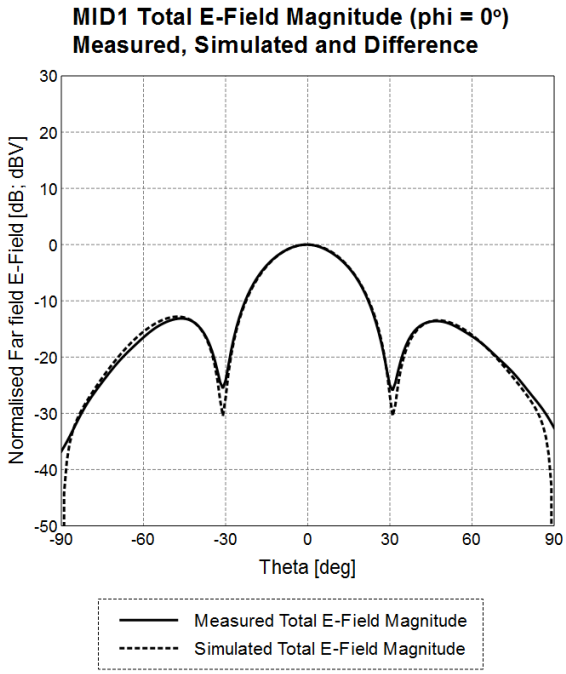
Another interesting avenue for future investigation is how to correct for distortions in the far-field pattern (see Fig. 4.2) when a failure element is detected.

ASTRON has been using an unsupervised learning method to cluster data from LOFAR. We recommend collaboration with ASTRON in future work, to see if there are other sources of data in the antenna array that can be used as training data for a machine-learning algorithm. For example, future work could apply machine learning to detect element failure from data such as the power spectral density, correlations and reflection coefficients of antenna arrays.

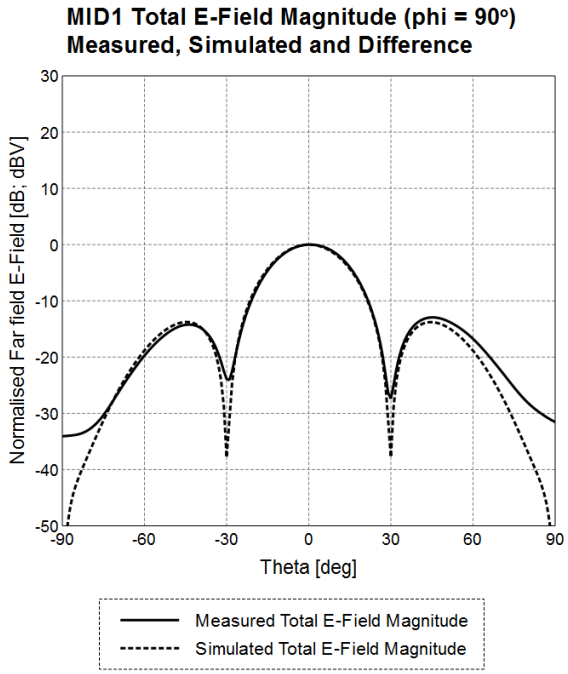
A Far-field measurements of various failure scenarios in the 16-element circular patch antenna array

A spherical near-field scanner in an anechoic chamber was used to measure the far-field patterns of each of the six MIDs in Table 5.3. To emulate an antenna failure, the relevant antenna's cable was detached from the antenna and the power combiner, and the port on the power combiner was terminated with a matched (50-Ω) load.

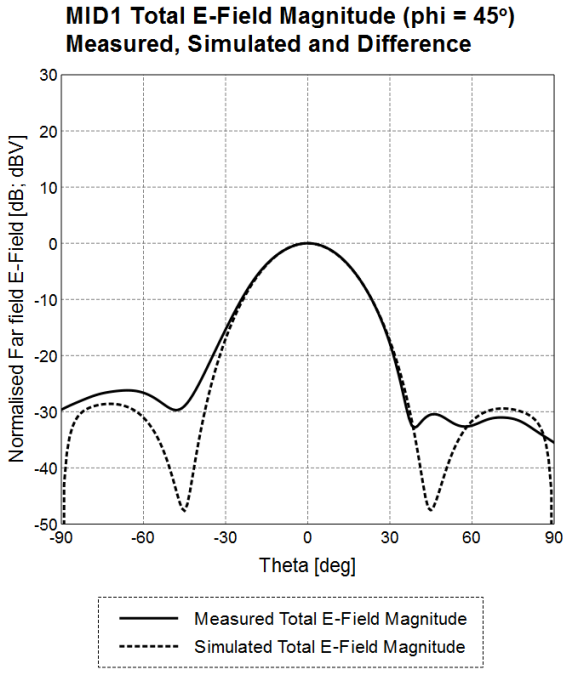
The principal and diagonal cuts of the simulated and measured far-field patterns of each MID are shown in Figs. A.1 to A.6 below.



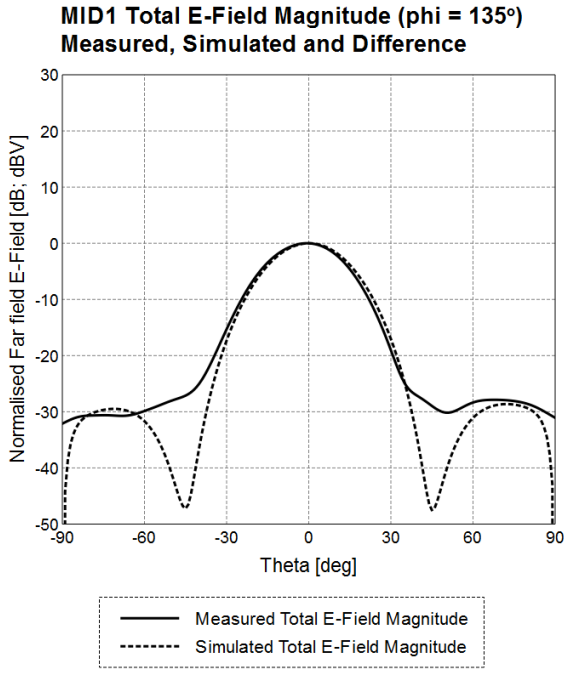
(A) $0°$ cut



(B) $90°$ cut

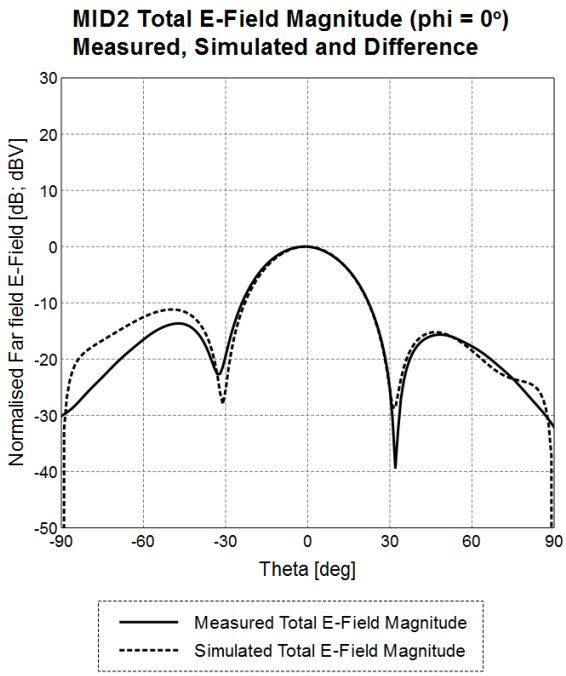


(C) $45°$ cut

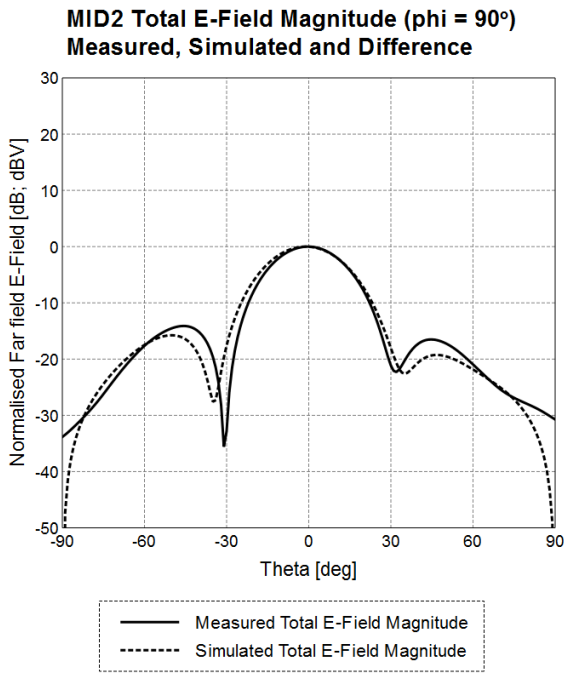


(D) $135°$ cut

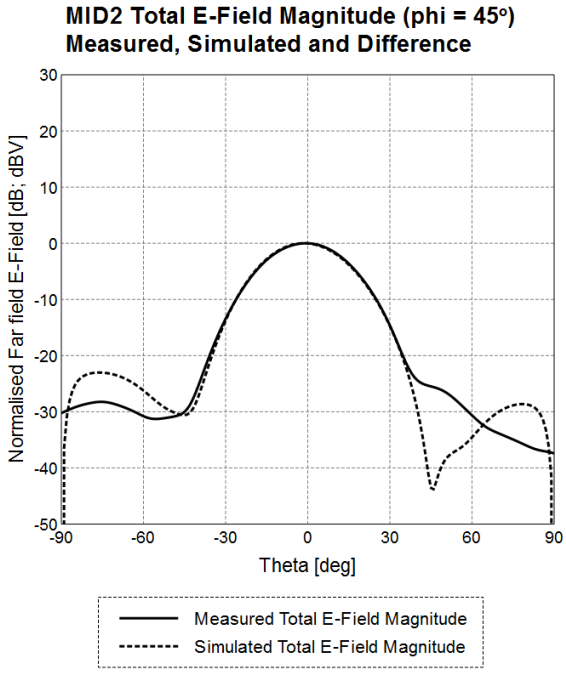
FIGURE A.1: The principal and diagonal cuts of MID 1. The measurements were taken at $f_o = 3.87$ GHz



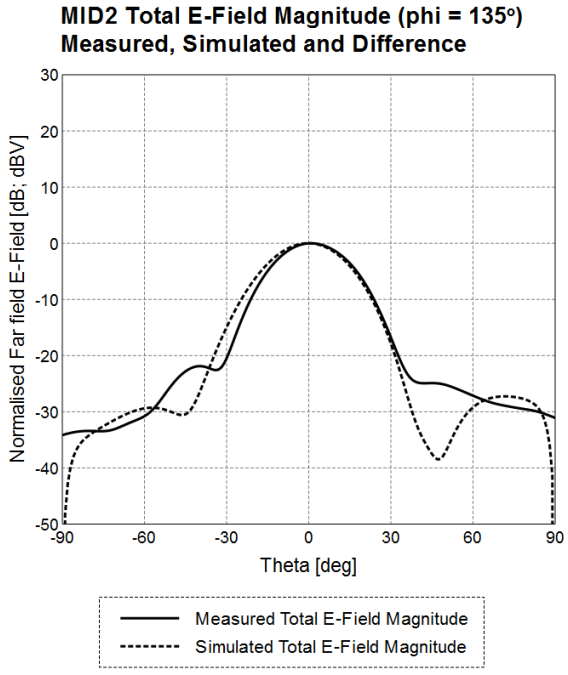
(A) $0°$ cut



(B) $90°$ cut

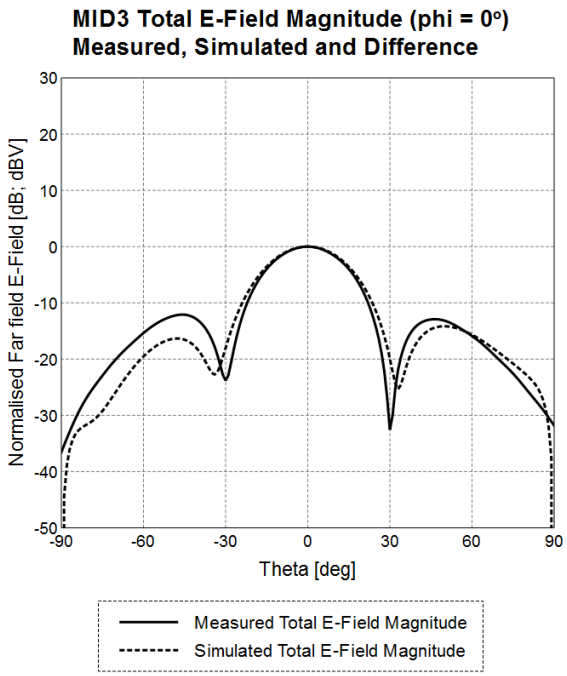


(C) $45°$ cut

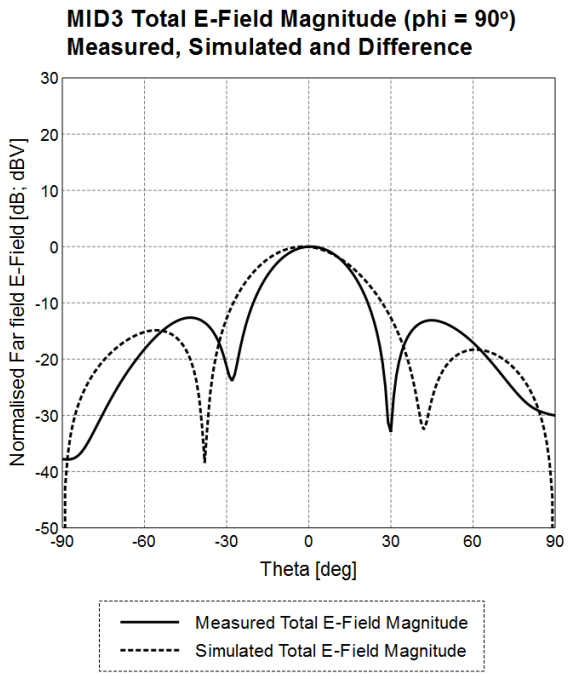


(D) $135°$ cut

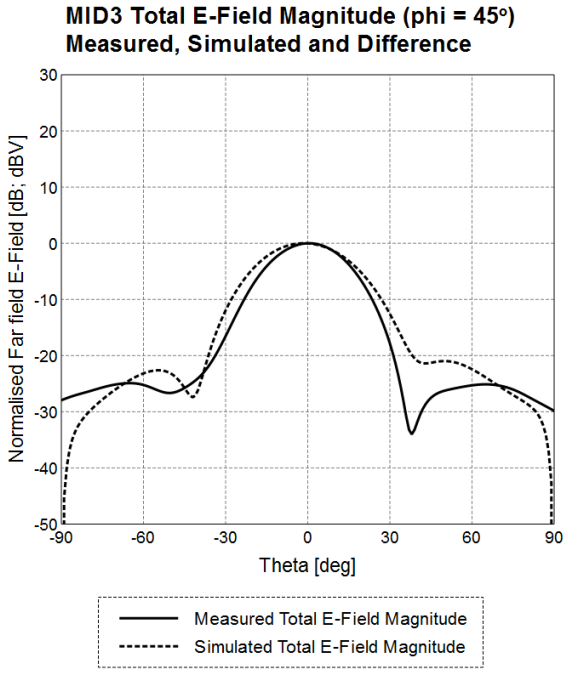
FIGURE A.2: The principal and diagonal cuts of MID 2. The measurements were taken at $f_o = 3.87$ GHz



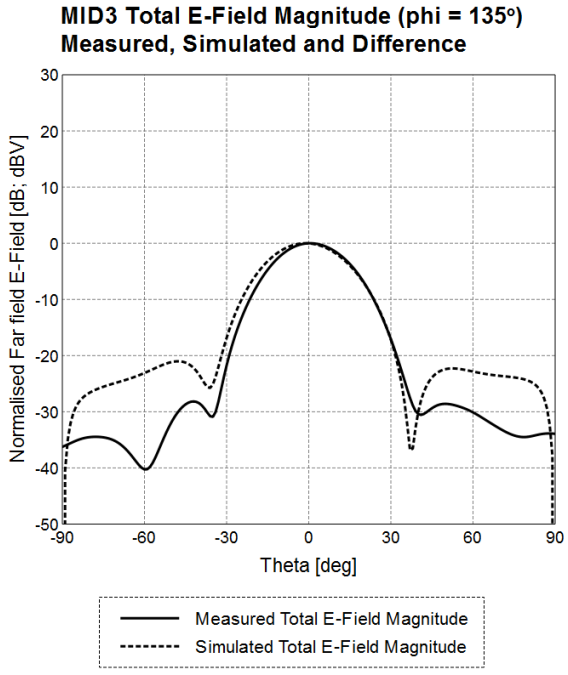
(A) $0°$ cut



(B) $90°$ cut

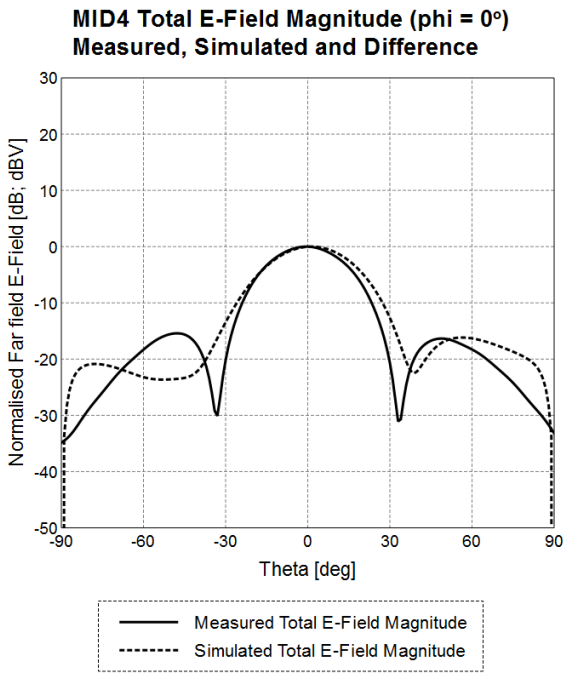


(C) $45°$ cut

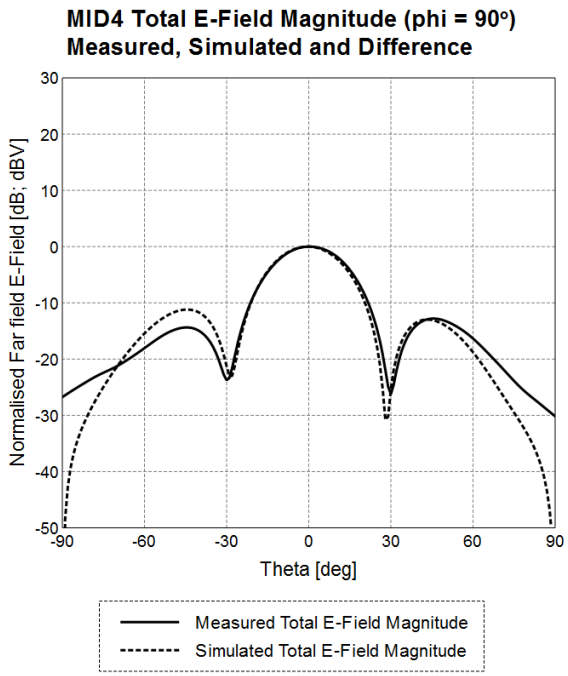


(D) $135°$ cut

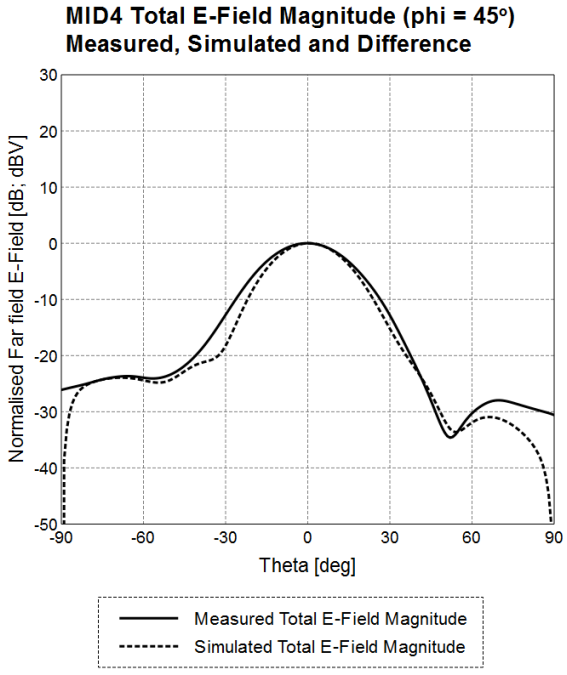
FIGURE A.3: The principal and diagonal cuts of MID 3. The measurements were taken at $f_o = 3.87$ GHz



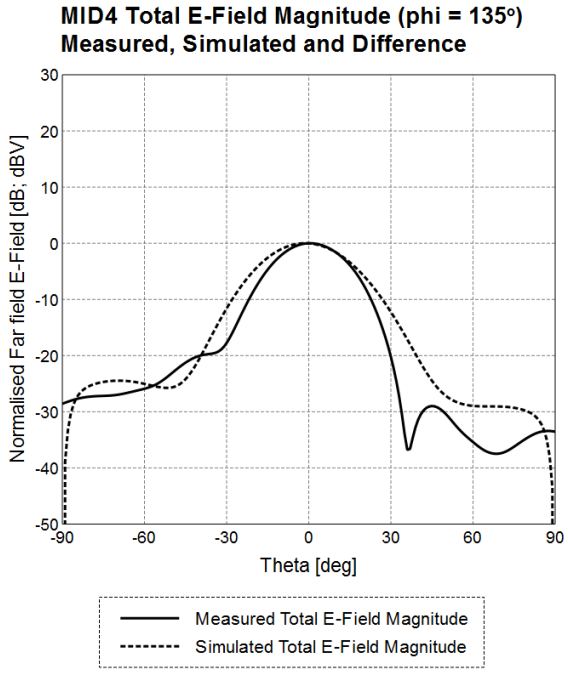
(A) 0° cut



(B) 90° cut

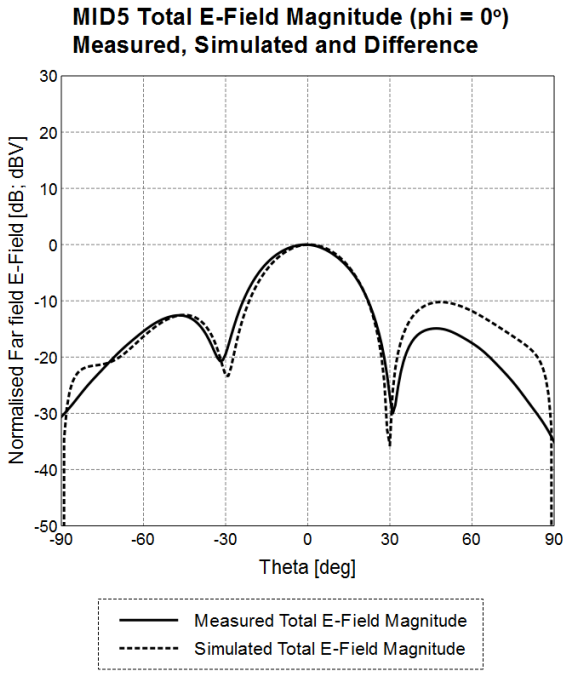


(C) 45° cut

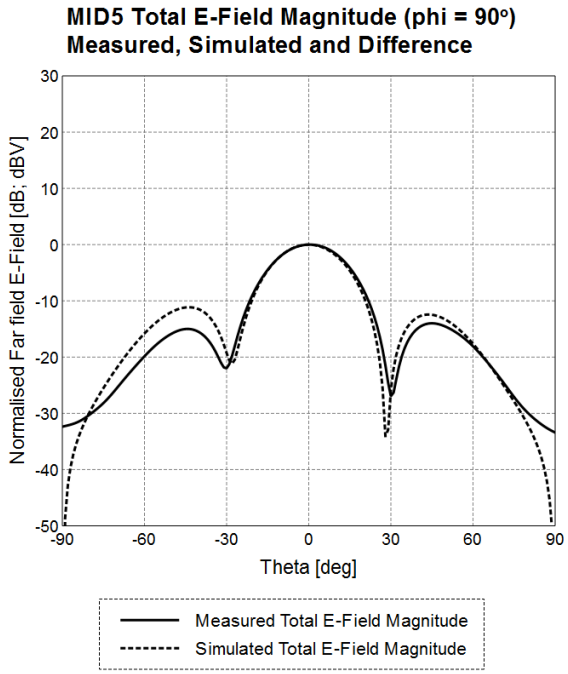


(D) 135° cut

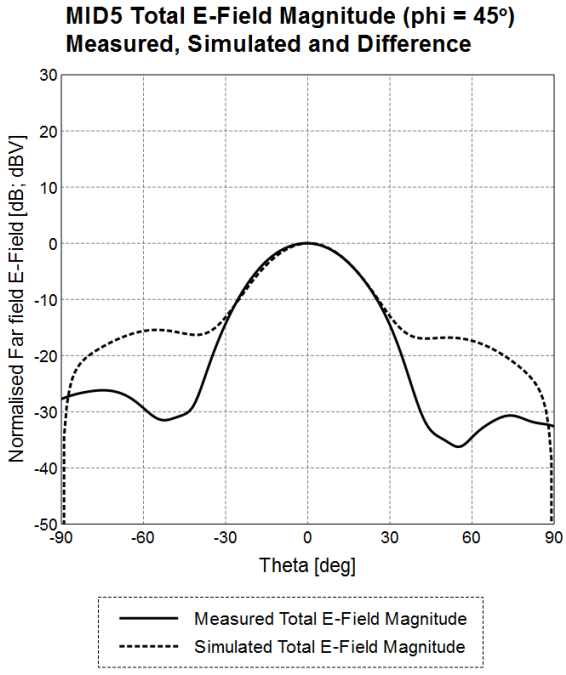
FIGURE A.4: The principal and diagonal cuts of MID 4. The measurements were taken at $f_o = 3.87$ GHz



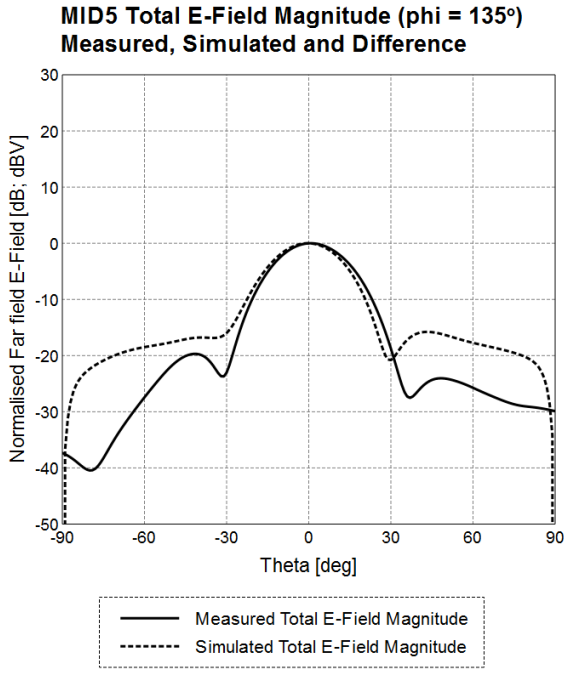
(A) 0° cut



(B) 90° cut

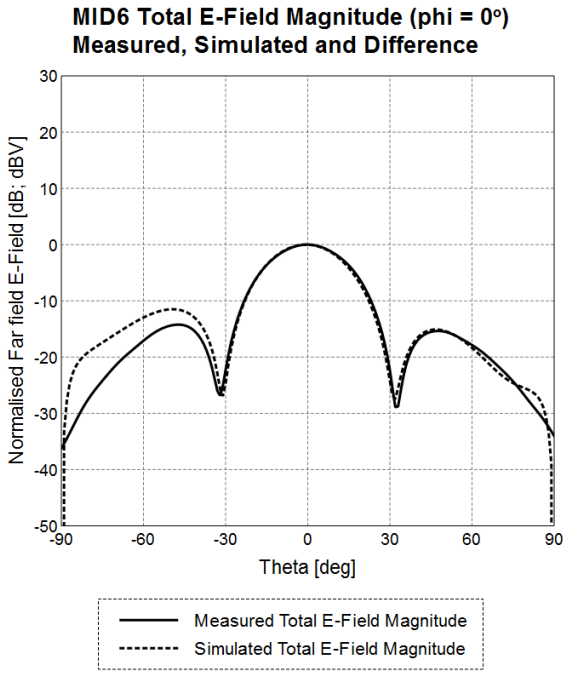


(C) 45° cut

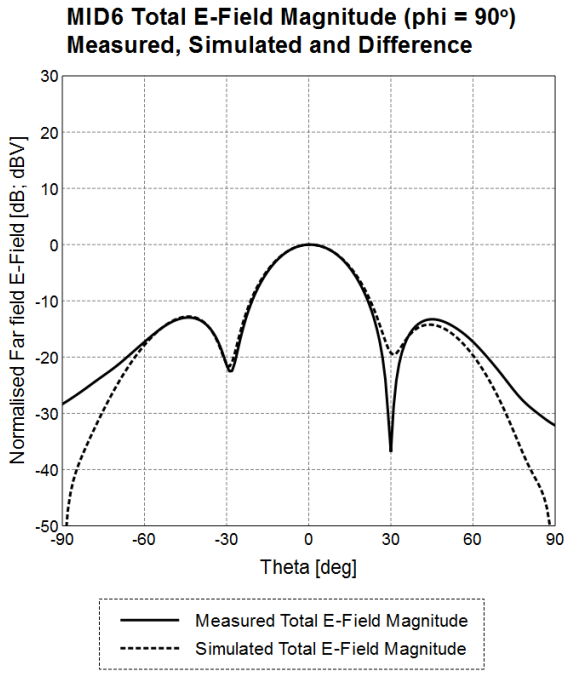


(D) 135° cut

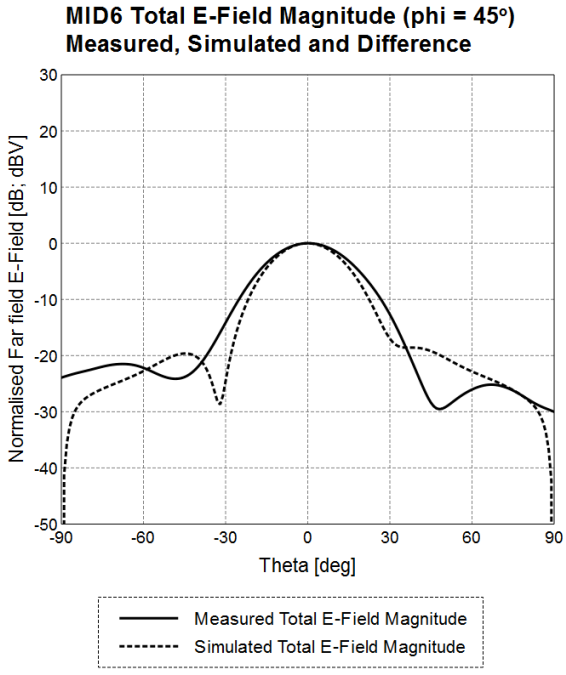
FIGURE A.5: The principal and diagonal cuts of MID 5. The measurements were taken at $f_o = 3.87$ GHz



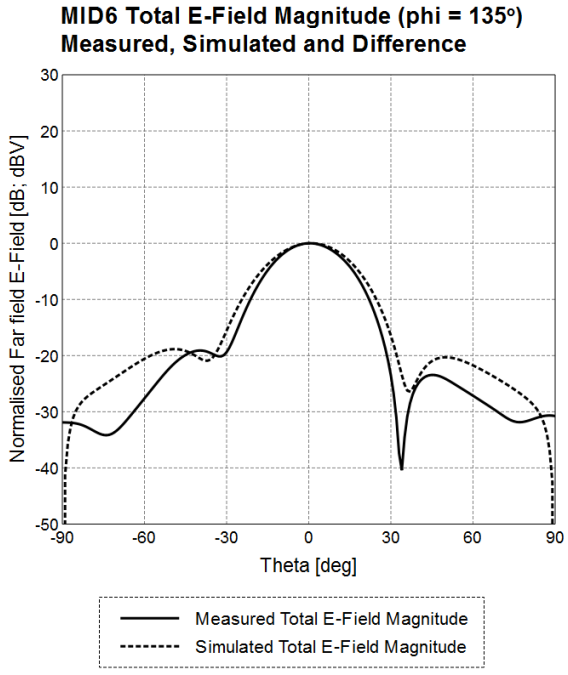
(A) 0° cut



(B) 90° cut



(C) 45° cut



(D) 135° cut

FIGURE A.6: The principal and diagonal cuts of MID 6. The measurements were taken at $f_o = 3.87$ GHz

Bibliography

[1] "IEEE Standard for Definitions of Terms for Antennas", *IEEE Std 145-2013 (Revision of IEEE Std 145-1993)*, pp. 1–50, 2014. DOI: 10.1109/IEEESTD.2014.6758443.

[2] G. W. Kant, P. D. Patel, S. J. Wijnholds, M. Ruiter, and E. van der Wal, "Embrace: A multi-beam 20,000-element radio astronomical phased array antenna demonstrator", *IEEE Transactions on Antennas and Propagation*, vol. 59, no. 6, pp. 1990–2003, 2011. DOI: 10.1109/TAP.2011.2122233.

[3] E. Geyh, T. Y. Sikina, T. R. Adhami, and J. J. Schuss, *Phased array radio frequency (rf) built-in-test equipment (bite) apparatus and method of operation therefor*, 1997. [Online]. Available: https://patents.google.com/patent/US5867123A/en.

[4] J. W. Locke, W. J. Haber, and P. A. Chiavacci, *Method and apparatus for mitigating array antenna performance degradation caused by element failure*, 1999. [Online]. Available: https://patents.google.com/patent/US6140976A/en.

[5] D. K. Vail, F. J. Tabor, D. P. Blom, and S. S. Wilson, *Phased array antenna including element control device providing fault detection and related methods*, 2001. [Online]. Available: https://patents.google.com/patent/US6573862B2/en.

[6] C. Simard and M. Drummy, *Method and system for transducer element fault detection for phased array ultrasonic instruments*, 2009. [Online]. Available: https://patents.google.com/patent/US8577629B2/en.

[7] P. J. Hall, *The square kilometre array: An engineering perspective*. Springer, 2005, vol. 2.

[8] J. A. Rodriguez, F. Ares, E. Moreno, and G. Franceschetti, "Genetic algorithm procedure for linear array failure correction", *Electronics Letters*, vol. 36, no. 3, pp. 196–198, 2000. DOI: 10.1049/el:20000236.

[9] R. J. Mailloux, "Array failure correction with a digitally beamformed array", *IEEE Transactions on Antennas and Propagation*, vol. 44, no. 12, pp. 1543–1550, 1996. DOI: 10.1109/8.546240.

[10] Beng-Kiong Yeo and Yilong Lu, "Array failure correction with a genetic algorithm", *IEEE Transactions on Antennas and Propagation*, vol. 47, no. 5, pp. 823–828, 1999. DOI: 10.1109/8.774136.

[11] T. J. Peters, "A conjugate gradient-based algorithm to minimize the sidelobe level of planar arrays with element failures", *IEEE Transactions on Antennas and Propagation*, vol. 39, no. 10, pp. 1497–1504, 1991. DOI: 10.1109/8.97381.

[12] C. Cortes and V. Vapnik, "Support vector machine", *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[13] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[14] M. de Vos, A. W. Gunst, and R. Nijboer, "The lofar telescope: System architecture and signal processing", *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1431–1437, 2009. DOI: 10.1109/JPROC.2009.2020509.

[15] M. á. van Haarlem, M. Wise, A. Gunst, G. Heald, J. McKean, J. Hessels, A. De Bruyn, R. Nijboer, J. Swinbank, R. Fallows, *et al.*, "Lofar: The low-frequency array", *Astronomy & astrophysics*, vol. 556, A2, 2013.

[16] P Dewdney, W Turner, R Millenaar, R McCool, J Lazio, and T Cornwell, "Ska1 system baseline design", *Document number SKA-TEL-SKO-DD-001 Revision*, vol. 1, no. 1, 2013.

[17] *LOFAR | Astron*, accessed 2019-11-20. [Online]. Available: https://www.astron.nl/telescopes/lofar.

[18] S. Johnston, R Taylor, M. Bailes, N. Bartel, C Baugh, M Bietenholz, C. Blake, R Braun, J Brown, S. Chatterjee, *et al.*, "Science with askap", *Experimental astronomy*, vol. 22, no. 3, pp. 151–273, 2008.

[19] R. Booth and J. Jonas, "An overview of the meerkat project", *African Skies*, vol. 16, p. 101, 2012.

[20] *ASTRON*, accessed 2019-11-01. [Online]. Available: https://www.astron.nl/.

[21] *SARAO - SKA SA*, accessed 2019-11-03. [Online]. Available: https://www.ska.ac.za/about/sarao.

[22] *Research Infrastructure Platforms - National Research Foundation*, accessed 2019-11-03. [Online]. Available: https://www.nrf.ac.za/research-platforms/national-facilities.

[23] *Organisational Structure - National Research Foundation*, accessed 2019-11-03. [Online]. Available: https://www.nrf.ac.za/about-nrf/organisational-structure.

[24] *SALT - The Southern African Large Telescope*, accessed 2019-11-03. [Online]. Available: https://www.salt.ac.za.

[25] L. Heystek, "Ska1-mid physical configuration coordinates", *Document number SKA-TEL-INSA-0000537 Revision 2*, vol. 1, no. 1, 2015.

[26] *Research Chairs - SKA SA*, accessed 2019-11-03. [Online]. Available: www.ska.ac.za/students/research-chairs.

[27] *Tenders and RFQ's - SKA SA*, accessed 2019-11-03. [Online]. Available: ska.ac.za/tenders.

[28] F. Harrou and M. N. Nounou, "Monitoring linear antenna arrays using an exponentially weighted moving average-based fault detection scheme", *Systems Science & Control Engineering*, vol. 2, no. 1, pp. 433–443, 2014.

[29] Kuan-Min Lee, Ruey-Shi Chu, and Sien-Chang Liu, "A built-in performance-monitoring/fault isolation and correction (pm/fic) system for active phased-array antennas", *IEEE Transactions on Antennas and Propagation*, vol. 41, no. 11, pp. 1530–1540, 1993. DOI: 10.1109/8.267353.

[30] B. Yeo and Y. Lu, "Expeditious diagnosis of linear array failure using support vector machine with low-degree polynomial kernel", *IET Microwaves, Antennas Propagation*, vol. 6, no. 13, pp. 1473–1480, 2012.

[31] R. Iglesias, F. Ares, M. Fernandez-Delgado, J. A. Rodriguez, J. Bregains, and S. Barro, "Element failure detection in linear antenna arrays using case-based reasoning", *IEEE Antennas and Propagation Magazine*, vol. 50, no. 4, pp. 198–204, 2008. DOI: 10.1109/MAP.2008.4653709.

[32] J. A. Rodriguez, M. Fernandez-Delgado, J. Bregains, R. Iglesias, S. Barro, and F. Ares, "A comparison among several techniques for finding defective elements in antenna arrays", in *The Second European Conference on Antennas and Propagation, EuCAP 2007*, 2007, pp. 1–8. DOI: 10.1049/ic.2007.1141.

[33] O. M. Bucci, A. Capozzoli, and G. D'Elia, "Diagnosis of array faults from far-field amplitude-only data", *IEEE Transactions on Antennas and Propagation*, vol. 48, no. 5, pp. 647–652, 2000. DOI: 10.1109/8.855482.

[34] D. L. Jones, K. Wagstaff, D. R. Thompson, L. D'Addario, R. Navarro, C. Mattmann, W. Majid, J. Lazio, R. Preston, and U. Rebbapragada, "Big data challenges for large radio arrays", in *2012 IEEE Aerospace Conference*, 2012, pp. 1–6. DOI: 10.1109/AERO.2012.6187090.

[35] D. R. Thompson, S. Burke-Spolaor, A. T. Deller, W. A. Majid, D. Palaniswamy, S. J. Tingay, K. L. Wagstaff, and R. B. Wayth, "Real-time adaptive event detection in astronomical data streams", *IEEE Intelligent Systems*, vol. 29, no. 1, pp. 48–55, 2014. DOI: 10.1109/MIS.2013.10.

[36] C. J. Wolfaardt, "Machine learning approach to radio frequency interference (rfi) classification in radio astronomy", PhD thesis, Stellenbosch: Stellenbosch University, 2016.

[37] M. L. James and L. P. Dubon, "An autonomous diagnostic and prognostic monitoring system for nasa's deep space network", in *2000 IEEE Aerospace Conference. Proceedings (Cat. No.00TH8484)*, vol. 2, 2000, 403–414 vol.2. DOI: 10.1109/AERO.2000.878248.

[38] A. Patnaik and C. Christodoulou, "Finding failed element positions in linear antenna arrays using neural networks", in *2006 IEEE Antennas and Propagation Society International Symposium*, 2006, pp. 1675–1678. DOI: 10.1109/APS.2006.1710883.

[39] Jing Miao, Bo Chen, Xiaolin Zhang, and Yang Chen, "An improved method of diagnosis of failed elements in arrays based on far-field radiation pattern", in *2016 Progress in Electromagnetic Research Symposium (PIERS)*, 2016, pp. 467–471. DOI: 10.1109/PIERS.2016.7734367.

[40] Nan Xu, C. G. Christodoulou, S. E. Barbin, and M. Martinez-Ramon, "Detecting failure of antenna array elements using machine learning optimization", in *2007 IEEE Antennas and Propagation Society International Symposium*, 2007, pp. 5753–5756. DOI: 10.1109/APS.2007.4396858.

[41] B. Choudhury, O. P. Acharya, and A. Patnaik, "Bacteria foraging optimization in antenna engineering: An application to array fault finding", *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 23, no. 2, pp. 141–148, 2013.

[42] D. J. Rochblatt, P. H. Richter, and T. Y. Otoshi, "A microwave performance calibration system for nasa's deep space network antennas .2. holography, alignment, and frequency stability", in *Tenth International Conference on Antennas and Propagation (Conf. Publ. No. 436)*, vol. 1, 1997, 150–155 vol.1. DOI: 10.1049/cp:19970227.

[43] H. Zhao, Y. Zhang, E. Li, A. Buonanno, and M. D'Urso, "Diagnosis of array failure in impulsive noise environment using unsupervised support vector regression method", *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 11, pp. 5508–5516, 2013. DOI: 10.1109/TAP.2013.2275750.

[44] A. Rawat, R. Yadav, and S. Shrivastava, "Neural network applications in smart antenna arrays: A review", *AEU-International Journal of Electronics and Communications*, vol. 66, no. 11, pp. 903–912, 2012.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[46] L. de Lange and D. J. Ludick, "Application of machine learning for antenna array failure analysis", in *2018 International Workshop on Computing, Electromagnetics, and Machine Intelligence (CEMi)*, IEEE, 2018, pp. 5–6.

[47] L. de Lange, D. J. Ludick, and T. L. Grobler, "Detecting failed elements in an arbitrary antenna array using machine learning", in *2019 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, 2019, pp. 1099–1103. DOI: 10.1109/ICEAA.2019.8879067.

[48] Altair, *Feko*, version FEKO 2018, May 3, 2018. DOI: 10.1098/rsta.1909.0016. [Online]. Available: https://altairhyperworks.com/product/FEKO.

[49] C. A. Balanis, "Antenna theory: A review", *Proceedings of the IEEE*, vol. 80, no. 1, pp. 7–23, 1992. DOI: 10.1109/5.119564.

[50] C. A. Balanis, *Antenna Theory: Analysis and Design 3rd Edition*. John Wiley & Sons, 2005, pp. 843–854.

[51] J. C. Maxwell, *A treatise on electricity and magnetism*. Oxford Univ. Press, 1873.

[52] *Karl Jansky and his Merry-go-Round - National Radio Astronomy Observatory*, accessed 2019-11-23. [Online]. Available: https://public.nrao.edu/gallery/karl-jansky-and-his-merrygoround/.

[53] J. Kraus, "Antennas since hertz and marconi", *IEEE Transactions on Antennas and Propagation*, vol. 33, no. 2, pp. 131–137, 1985, ISSN: 1558-2221. DOI: 10.1109/TAP.1985.1143550.

[54] S. Silver, *Microwave antenna theory and design*, 19. Iet, 1984.

[55] D. Systemes, *Antenna magus*, version 2019.1, Jan. 4, 2019. [Online]. Available: https://www.cst.com/products/antennamagus.

[56] R. Nan, D. Li, C. Jin, Q. Wang, L. Zhu, W. Zhu, H. Zhang, Y. Yue, and L. Qian, "The five-hundred-meter aperture spherical radio telescope (fast) project", *International Journal of Modern Physics D*, vol. 20, no. 06, pp. 989–1024, 2011.

[57] N. Dennis, *World's largest radio telescope will search for dark matter, listen for aliens*, accessed 2019-11-23, 2016. [Online]. Available: https://www.sciencemag.org/news/2016/09/world-s-largest-radio-telescope-will-search-dark-matter-listen-aliens.

[58] C. A. Balanis, *Modern Antenna Handbook*. John Wiley & Sons, 2011.

[59] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[60] G. Bonaccorso, *Mastering Machine Learning Algorithms: Expert techniques to implement popular machine learning algorithms and fine-tune your models*. Packt Publishing Ltd, 2018.

[61] R. Bellman, "Dynamic programming and lagrange multipliers", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 10, p. 767, 1956.

[62] B. Dustin, *Introduction to Support Vector Machines*, accessed 2019-09-03, 2002. [Online]. Available: http://www.work.caltech.edu/~boswell/IntroToSVM.pdf.

[63] N. Cristianini, J. Shawe-Taylor, *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[64] B. Schölkopf, C. J. Burges, A. J. Smola, *et al.*, *Advances in kernel methods: support vector learning*. MIT press, 1999.

[65] C. J. Burges and B. Schölkopf, "Improving the accuracy and speed of support vector machines", in *Advances in neural information processing systems*, 1997, pp. 375–381.

[66] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for machine learning*, 2018.

[67] J Mercer, *Functions of positive and negative type and their connection with the theory of integral equations*, 1909.

[68] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's razor", *Information processing letters*, vol. 24, no. 6, pp. 377–380, 1987.

[69] J. Nalepa and M. Kawulok, "Selecting training sets for support vector machines: A review", *Artificial Intelligence Review*, vol. 52, no. 2, pp. 857–900, 2019.

[70] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012, vol. 329.

[71] S. Menard, *Applied logistic regression analysis*. Sage, 2002, vol. 106.

[72] S. Baluja, "Evolution of an artificial neural network based autonomous land vehicle controller", *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 26, no. 3, pp. 450–463, 1996.

[73] B. K. Wong and Y. Selvi, "Neural network applications in finance: A review and analysis of literature (1990–1996)", *Information & Management*, vol. 34, no. 3, pp. 129–139, 1998.

[74] A. S. Weigend, *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018.

[75] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce english text", *Complex systems*, vol. 1, no. 1, pp. 145–168, 1987.

[76] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[77] L. Zhang and B. Zhang, "A geometrical representation of mcculloch-pitts neural model and its applications", *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 925–929, 1999.

[78] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, 10. Springer series in statistics New York, 2001, vol. 1.

[79] R. Corporation, "RO4000 ® Series High Frequency Circuit Materials Datasheet", pp. 1–4,

[80] H. J. du Toit, D. I. L. de Villiers, and R. D. Beyers, "A simple low loss partially-filled 16-way radial power combiner", in *2019 IEEE MTT-S International Microwave Symposium (IMS)*, 2019, pp. 440–443. DOI: 10.1109/MWSYM.2019.8700892.

[81] *Spherical Near-Field Scanner Systems*, accessed 2019-11-13. [Online]. Available: https://www.nsi-mi.com/products/system-solutions/near-field-systems/spherical-near-field-scanner-systems.