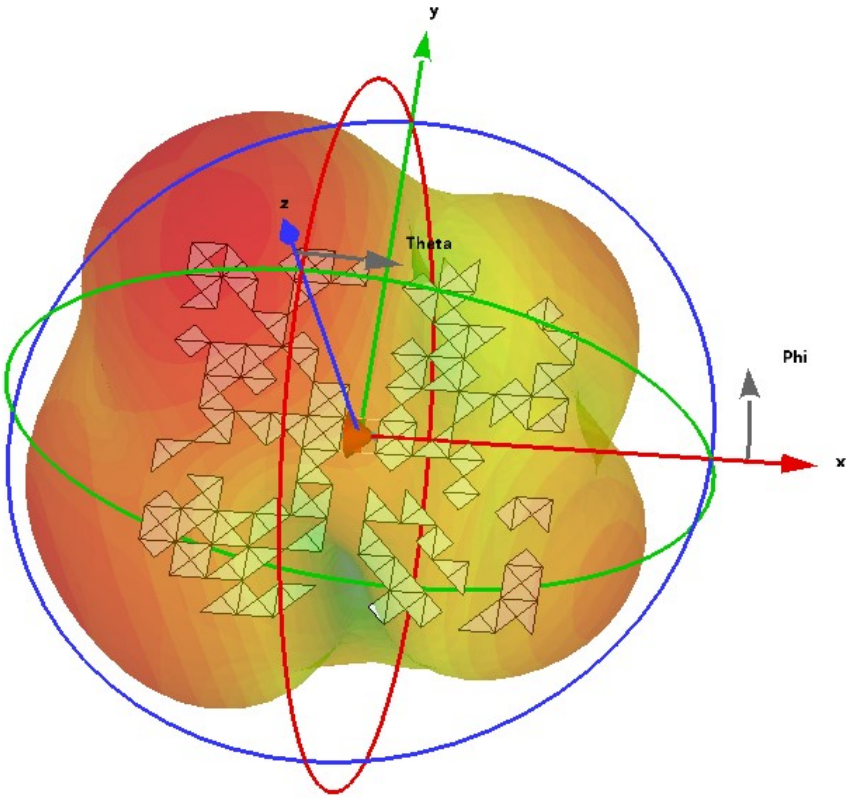
CHALMERS
UNIVERSITY OF TECHNOLOGY



Machine Learning-Assisted Synthesis of Filtering Antennas Using a Fast Method of Moments Code

Master's thesis in Master program Wireless, Photonics and Space

FITIM MAXHARRAJ

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Machine Learning-Assisted Synthesis of Filtering Antennas Using a Fast Method of Moments Code

FITIM MAXHARRAJ

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Communications, Antennas, and Optical Networks
Antenna Systems
Chalmers University of Technology
Gothenburg, Sweden 2024

Machine Learning-Assisted Synthesis of Filtering Antennas Using a Fast Method of Moments Code

FITIM MAXHARRAJ

Examiner: Prof. Rob Maaskant, Chalmers
Supervisor: Prof. Rob Maaskant, Chalmers
Co-Supervisor: Dr. Martin Sjödin, Ericsson

Master's Thesis 2024
Department of Electrical Engineering
Division of Communications, Antennas, and Optical Networks
Antenna Systems
Chalmers University of Technology
SE-412 96 Gothenburg

Cover: Visualization of radiation pattern of a ML-model antenna.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Machine Learning-Assisted Synthesis of Filtering Antennas Using a Fast Method of Moments Code

FITIM MAXHARRAJ
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis investigates the integration of machine learning algorithms with electromagnetic simulations as a novel strategy to optimize antenna designs, thereby significantly enhancing simulation speed and performance in modern antenna systems. The study utilizes an in-house Method of Moments (MoM) software, CAESAR, to rapidly generate a comprehensive dataset of antennas. This is achieved in a time-efficient manner by removing the contribution of Rao-Wilton-Glisson (RWG) basis functions in the MoM matrix. A convolutional neural network (CNN) was selected for its superior pattern recognition capabilities, enabling the model to accurately predict the scattering parameters and gain of various antennas. Furthermore, a genetic algorithm was employed to optimize the antenna designs in conjunction with the trained CNN model.

The research includes a thorough validation of the model's accuracy and reliability, assessed through mean squared error metrics and extensive simulations. Remarkably, the in-house software CAESAR exhibited exceptional efficiency, surpassing commercial software CST coupled with TCST Interface by over 2000%. The results demonstrate the efficiency of combining machine learning with electromagnetic simulations in improving antenna design processes, potentially setting a new standard for future advancements in methodology in antenna design.

Keywords: Machine Learning, Filtering Antennas, Method of Moments, Electromagnetic Simulation, Optimization.

Acknowledgements

I would like to extend my sincere thanks to Prof. Rob Maaskant for his invaluable mentorship and guidance throughout my Bachelor's Thesis and now my Master's Thesis. Without Rob's support, this thesis would not have been possible.

I am also deeply grateful to my supervisor, Dr. Martin Sjödin, for his exceptional expertise in machine learning.

Lastly, I would like to express my profound gratitude to my family and my wife, Flaureta Rexhaj, for their unwavering love and support throughout my academic journey.

Fitim Maxharraj, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AI	Artificial Intelligence
ANN	Artificial Neural Network
BPF	Band-Pass Filter
CEM	Computational Electromagnetics
CNN	Convolutional Neural Network
EFIE	Electric Field Integral Equation
FEM	Finite Element Method
FDTD	Finite Difference Time domain
GA	Genetic Algorithm
GPU	Graphics Processing Unit
MAPE	Mean Absolute Percentage Error
MFIE	Magnetic Field Integral Equation
ML	Machine Learning
MoM	Methods of Moments
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
RWG	Rao-Wilton-Glisson
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
PEC	Perfect Electrical Conductor
PMCHWT	Poggio Miller–Chang–Harrington–Wu–Tsai

Contents

List of Figures

List of Tables

1
Introduction

In this chapter, the background, aim, and outline of this thesis are presented.

1.1 Background

As industries continue to innovate and develop cutting-edge technologies, the ability to accurately predict electromagnetic phenomena quickly and reliably plays a crucial role in ensuring the success of various engineering tasks. From the design of advanced communication systems to the development of sophisticated radar systems, accurate simulations of electromagnetics are fundamental to achieving optimal performance.

The increasing demands for high-performance wireless communication systems have driven the need for efficient and optimal design methodologies in the fields of electromagnetics and antenna design. The combination of machine learning algorithms and electromagnetic simulations offers a novel approach to optimizing antenna designs, thereby enhancing their capabilities in modern communication systems.

Traditional electromagnetic simulation techniques, while efficient, often encounter limitations in terms of computational efficiency. As the complexity of electromagnetic systems continues to grow, the computational resources required to accurately analyze and model these systems also increase. Consequently, there arises a need for innovative methodologies and tools that can make the simulation process more efficient and less time-consuming, enabling researchers and engineers to obtain accurate results within a reasonable timeframe.

Over the past decade, machine learning has become increasingly utilized in fields such as financial markets, marketing, and many others to predict trends. Surprisingly, in computer-aided antenna design, machine learning algorithms are used less frequently. If applied, they could open up new, unthinkable possibilities to explore designs that humans have never imagined.

There has been a study using neural networks for impedance matching [1]. However, the dataset used to train the models was not sufficiently large, with only 377 samples, which took approximately 90 minutes to generate. Another study utilized machine learning successfully to enhance the efficiency and accuracy of a multi-band

rectangular spiral-shaped patch antenna [2]. In [3], machine learning was employed to predict antenna dimensions for a rectangular microstrip patch antenna based on performance requirements.

In antenna arrays, the optimal array lattice was synthesized using neural networks in [4]. In [5], a neural network was utilized to compute the phases of the antenna array elements to synthesize the radiation pattern. Neural networks have also been successfully employed for beamforming [6], [7].

The utilization of genetic algorithms was investigated for designing a highly directional array [8]. In one example presented in [8], a single array design took over 270 hours to complete.

In response to these challenges, the utilization of a fast Method of Moments code, together with machine learning techniques, emerges as a promising approach to enhance the efficiency of electromagnetic simulations. By leveraging advanced computational electromagnetics and machine learning algorithms, researchers and engineers can develop robust frameworks capable of rapidly simulating complex electromagnetic phenomena while maintaining high levels of accuracy and reliability. Another advantage is that one may synthesize antenna solutions that would be nearly impossible to achieve using traditional design approaches. This makes neural networks particularly attractive due to their ability to solve complex optimization problems in a matter of minutes rather than hours, leading to smaller footprint in terms of environmental impact.

1.2 Aim

The aim of this thesis is to implement a fast Method of Moments code for efficient electromagnetic simulations, along with integrating a machine learning model to optimize the impedance and radiation characteristics of an antenna.

Additionally, the aim includes:

- Develop a comprehensive understanding of filtering antennas and their design principles, and investigate and identify the key parameters affecting their performance.
- Develop a comprehensive understanding of the Method of Moments and the in-house software CAESAR, and write scripts to mesh an antenna.
- Develop a database of simulated designs, including performance metrics, utilizing the fast Method of Moment code.
- Employ machine learning algorithms, such as neural networks and genetic algorithms, to analyze the relationship between design parameters and perfor-

mance metrics.

- Train the machine learning model using the generated database, and use the model to optimize antenna design configurations with a genetic algorithm.
- Validate the accuracy and reliability of the machine learning models through additional simulations.
- Evaluate the efficiency and effectiveness of the machine learning-assisted synthesis approach compared to traditional design methods.

1.3 Outline

The outline of this thesis is as follows. In Chapter 2, a brief theoretical background on filtering antennas is provided. Furthermore, an in-depth exploration of computational electromagnetics is presented, focusing on the method of moments starting from Maxwell's equations to equivalence principles, electric field integral equation, and extending to the choice of basis functions and the reaction integral. Additionally, the singularity subtraction method and the MoM matrix are described. Chapter 2 also introduces machine learning, specifically convolutional neural networks, and explains the concepts of layers, activation functions, optimizers, loss functions, and data augmentation in machine learning. Finally, a genetic algorithm is introduced.

In Chapter 3, the method used in this thesis is described. The process of meshing the antenna is explained, along with the method for removing the contribution of the RWG basis functions of a triangle on the meshed antenna. Additionally, the generation of the dataset on the Chalmers cluster Vera is detailed, including the concept of data augmentation. Chapter 3 also includes a list of the parameters used for training the model, an overview of the model architecture, and the model optimization step.

Chapter 4 presents the results, including the antenna gain and port scattering parameters, of the meshed antenna on a full-metal antenna and one randomized antenna, which is not included in the training set of the machine learning model. The dataset generated on the cluster is presented along with the machine learning model's losses. Predictions on the gain and scattering parameters of the full-metal antenna and the randomized antenna are shown, along with two accurately optimized antennas with the same desired S-parameters and gain, and one with less accurate predictions. Lastly, a dual-resonance antenna and its radiation pattern are presented.

Finally, in Chapter 5, the conclusions and future work are presented.

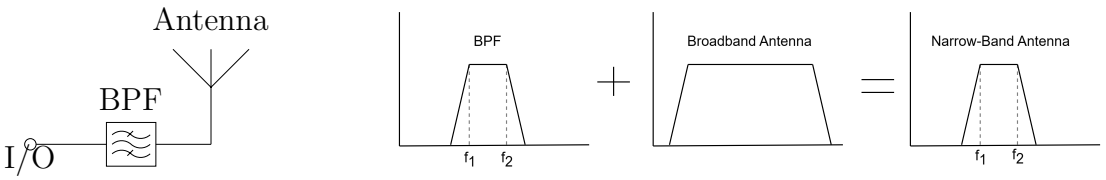
2

Theoretical Framework

In this chapter, the theoretical background of filtering antennas, computational electromagnetics, and machine learning used in this thesis are described.

2.1 Filtering Antennas

Filtering antennas integrate the functionalities of both filters and antennas into a single device. They can serve as transmitters and/or receivers while simultaneously providing frequency-selective filtering capabilities. The operation of a filtering antenna is achieved through the electromagnetic interaction of the antenna structure with the distributed filtering structure, enabling the selective transmission or reception of electromagnetic waves within a designated frequency band [9]. An illustration of the filtering antenna is shown in Fig. 2.1(a) and the frequency filtering is illustrated in Fig. 2.1(b).



(a) Antenna with a band-pass filter.

(b) Band-pass filter added to a broadband antenna.

Figure 2.1: (a) shows an antenna + external filter. (b) Shows the output of (a).

Filtering antennas work as conventional antennas by capturing incoming electromagnetic waves. Their design efficiently captures the incoming wave across the predefined frequency band, as dictated by the application requirements. The filtering component in a filtering antenna differentiates between the desired and undesired frequencies. The filtering is achieved through various structures, such as resonators or transmission lines, which exhibit frequency-dependent responses [9].

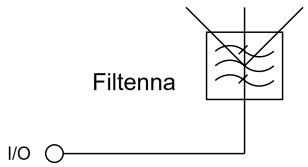


Figure 2.2: A illustration of an filtenna.

The benefits of truly merging antenna and filter functionalities, as illustrated in Fig. 2.2, are that filtennas can precisely control signal frequencies, enabling the isolation of specific frequency bands for mitigating interference. Filtennas can be compact and lightweight, suitable for applications with spatial constraints, which in turn reduces overall system costs, making them economically attractive [9].

In order to analyze, design, and optimize an filtenna, a computational electromagnetics tool is needed. In this thesis, the in-house computational electromagnetics software CAESAR is used [10].

2.2 Computational Electromagnetics

Computational Electromagnetics (CEM) is an interdisciplinary field that combines electromagnetic theory, numerical techniques, and computational science. It provides a sophisticated framework for analyzing and simulating electromagnetic phenomena. Computational electromagnetics is employed to address complex electromagnetic problems, involving the formulation and manipulation of mathematical models derived from Maxwell's equations and the fundamental principles of electromagnetism [11]. Through these models, computational electromagnetics enables the precise analysis and simulation of real-world electromagnetic phenomena.

The necessity of computational electromagnetics arises from the inherent complexity found in many real-world electromagnetic problems. Traditional analytical methods often struggle to address such challenges, limited by complex geometries, material properties and boundary conditions [12]. Computational electromagnetics provides a robust framework for overcoming these challenges, enabling engineers and researchers to design, optimize and understand electromagnetic devices and systems with exceptional accuracy and efficiency. Additionally, it offers a cost-effective and time-saving alternative to experimental methods, accelerating the analysis, prototyping and refinement of electromagnetic designs.

These field models undergo discretization, transforming field functions formulations into finite sets of field equations that can be used in numerical computations. Various numerical discretization techniques, such as finite difference (FDTD), finite element (FEM), and method of moments (MoM), are then used to solve the discretized equations and approximate the electromagnetic field quantities [13]. These methods have their strengths and weaknesses, making them suitable for different electromagnetic problems and applications. In this thesis, the commercial computational electromagnetic software CST Studio Suite is used for comparison with the in-house method of moments software CAESAR [10].

2.2.1 Method Of Moments

The method of moments is a numerical technique widely used in computational electromagnetics, particularly well-suited for problems involving open boundaries, such as radiation and scattering from complex objects. It offers advantages such as

accuracy and efficiency, especially for metal-only problems and homogeneous media. However, it may require significant computational resources for problems with dense meshes due to fine geometrical features and/or electrically large problems [14].

In the next subsections, starting from Maxwell's equations and the equivalence principle, which is a tool to replace complex sources and fields with simpler, equivalent sources on a hypothetical surface, making it easier to analyze, we will derive the electric field integral equation (EFIE). The EFIE is used to calculate the surface currents induced by electromagnetic waves on conductive objects. This equation is then discretized to facilitate the calculation of the surface currents.

2.2.1.1 Maxwell's Equation

Maxwell's equations are a set of four fundamental equations in electromagnetism that describe the behavior of the electric field and the magnetic field. These equations yield a unique solution if the boundary conditions, the permittivity ε, and the permeability μ are known [15].

The four fundamental Maxwell's equations in the frequency domain for fields generated by an electric current $\boldsymbol{J}$ are defined by

$$\boldsymbol{\nabla} \times \boldsymbol{E} = -j\omega\mu\boldsymbol{H} \tag{2.1a}$$
$$\boldsymbol{\nabla} \times \boldsymbol{H} = j\omega\varepsilon\boldsymbol{E} + \boldsymbol{J} \tag{2.1b}$$
$$\boldsymbol{\nabla} \cdot \boldsymbol{E} = \frac{\rho_\mathrm{e}}{\varepsilon} \tag{2.1c}$$
$$\boldsymbol{\nabla} \cdot \boldsymbol{H} = 0 \tag{2.1d}$$

where ρ_e is the electric charge density.

Since $\boldsymbol{\nabla} \times \boldsymbol{H} = 0$, the magnetic field $\boldsymbol{H}$ can be expressed in terms of a magnetic vector potential $\boldsymbol{H} = \boldsymbol{\nabla} \times \boldsymbol{A}$.

By substituting the magnetic vector potential, $\boldsymbol{\nabla} \times \boldsymbol{H} = 0$ in (2.1), and using the identity $\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times \boldsymbol{A} = \boldsymbol{\nabla}(\boldsymbol{\nabla} \cdot \boldsymbol{A} - \boldsymbol{\nabla^2}\boldsymbol{A})$, one can arrive at

$$\boldsymbol{E} = -j\omega\boldsymbol{A} - \boldsymbol{\nabla}\phi_\mathrm{e} \tag{2.2a}$$
$$\boldsymbol{H} = \frac{1}{\mu}\boldsymbol{\nabla} \times \boldsymbol{A} \tag{2.2b}$$

where

$$\boldsymbol{A} = \mu \int_{V'} \boldsymbol{J}(\boldsymbol{r}')G(\boldsymbol{r} - \boldsymbol{r}')\,\mathrm{d}V' \tag{2.3a}$$
$$\phi_\mathrm{e} = -\frac{1}{j\omega\varepsilon} \int_{V'} \boldsymbol{\nabla}' \cdot \boldsymbol{J}(\boldsymbol{r}')G(\boldsymbol{r} - \boldsymbol{r}')\,\mathrm{d}V' \tag{2.3b}$$

and where the scalar Green's function is defined by

$$G(\boldsymbol{r} - \boldsymbol{r}') = \frac{e^{-jkR}}{4\pi R} \tag{2.4}$$

where $R = |\boldsymbol{r} - \boldsymbol{r}'|$.

The concept of the vector potential $\boldsymbol{A}$ and the scalar potential ϕ_e is used to express the electric field in integral form, which will later be discretized. Before continuing, one needs to understand the boundary conditions, which will be determined using the equivalence principle.

2.2.1.2 Equivalence Principles

The initial step in deriving the boundary conditions involves defining two distinct regions, denoted as region one and region two in Fig. 2.3, where electric and magnetic fields are present.

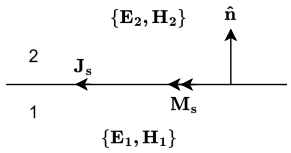


Figure 2.3: Boundary between two regions.

The boundary conditions between these two regions are defined by

$$\hat{\boldsymbol{n}} \times \boldsymbol{E}_2 - \hat{\boldsymbol{n}} \times \boldsymbol{E}_1 = -\boldsymbol{M}_s \tag{2.5a}$$

$$\hat{\boldsymbol{n}} \times \boldsymbol{H}_2 - \hat{\boldsymbol{n}} \times \boldsymbol{H}_1 = \boldsymbol{J}_s \tag{2.5b}$$

where $\boldsymbol{J}_s$ is the electric surface current density and $\boldsymbol{M}_s$ is the equivalent magnetic surface current density.

By applying boundary conditions and employing the superposition of both internal and external equivalents, as illustrated in Fig 2.4, the tangential component of $\boldsymbol{E}_2$ external to S must be equal to the tangential component of $\boldsymbol{E}_1$ internal to S to ensure continuity of the fields across S. This can be expressed mathematically as

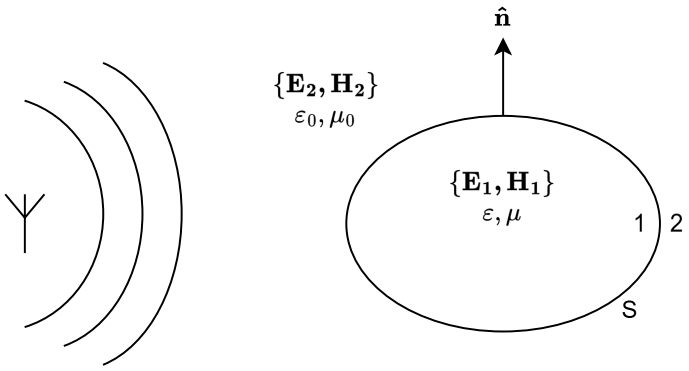
$$\boldsymbol{E}_2\Big|_{\tan} = \boldsymbol{E}_1\Big|_{\tan} \tag{2.6}$$

and where $\boldsymbol{E}_2 = \boldsymbol{E}^i + \boldsymbol{E}^s$ yields

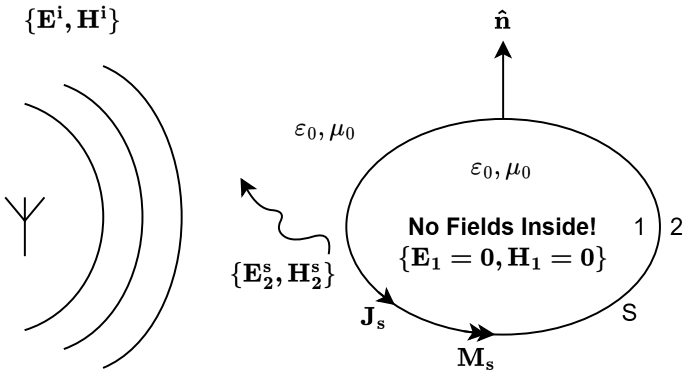
$$\hat{\boldsymbol{n}} \times \boldsymbol{E}_2^s(\boldsymbol{J}_s, \boldsymbol{M}_s) + \hat{\boldsymbol{n}} \times \boldsymbol{E}_1(\boldsymbol{J}_s, \boldsymbol{M}_s) = -\hat{\boldsymbol{n}} \times \boldsymbol{E}^i \tag{2.7a}$$

$$\hat{\boldsymbol{n}} \times \boldsymbol{H}_2^s(\boldsymbol{J}_s, \boldsymbol{M}_s) + \hat{\boldsymbol{n}} \times \boldsymbol{H}_1(\boldsymbol{J}_s, \boldsymbol{M}_s) = -\hat{\boldsymbol{n}} \times \boldsymbol{H}^i \tag{2.7b}$$

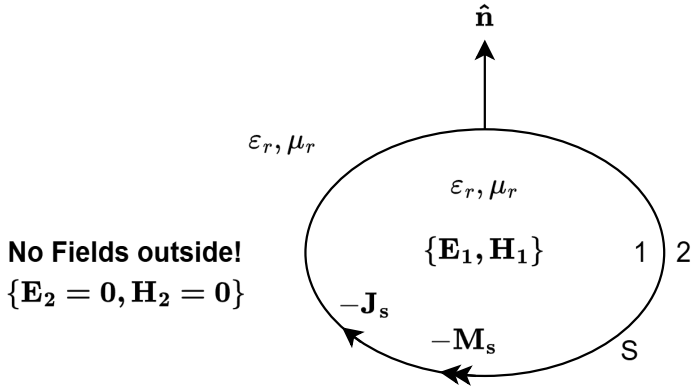
This formulation is known as the Poggio-Miller-Chang-Harrington-Wu-Tsai (PMCHWT) [16] formulation and is defined for dielectric interfaces. In this formulation, the excitation $\boldsymbol{E}^i$ and $\boldsymbol{H}^i$ are known, while the surface current density $\boldsymbol{J}_s$ and the equivalent magnetic current density $\boldsymbol{M}_s$ are unknown. The electric field requires the evaluation of an integral over the unknown currents, which will be explained in section 2.2.1.3.



(a) Original Problem

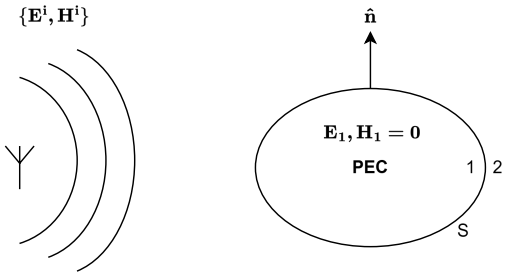


(b) External Equivalent of (a)

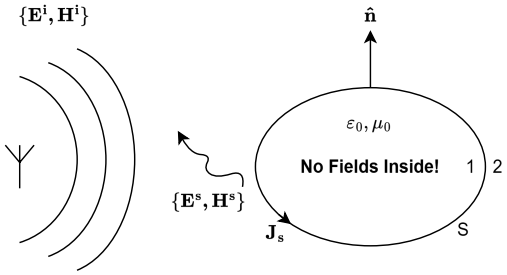


(c) Internal Equivalent of (a)

Figure 2.4: (a) shows the original problem. (b) shows the external equivalent problem of (a) and (c) shows the internal equivalent problem of (a).



(a) PEC Problem



(b) Physical equivalent of (a)

Figure 2.5: (a) shows the original problem. (b) shows the physical equivalent problem of (a).

The antennas used in this thesis are metal-only structures and can be modeled as perfect electric conductors (PECs). Applying the interface boundary condition in (2.5) to a PEC, as illustrated in Fig. 2.5, where the internal fields are zero and there are no magnetic currents [17], defines the boundary conditions as

$$\hat{\boldsymbol{n}} \times \boldsymbol{E}_2 = 0 \tag{2.8a}$$

$$\hat{\boldsymbol{n}} \times \boldsymbol{H}_2 = \boldsymbol{J}_s \tag{2.8b}$$

and because $\boldsymbol{E}_2 = \boldsymbol{E}^i + \boldsymbol{E}^s$, we can write

$$\hat{\boldsymbol{n}} \times \boldsymbol{E}^i + \hat{\boldsymbol{n}} \times \boldsymbol{E}^s(\boldsymbol{J}_s) = 0 \tag{2.9}$$

by using the identity $\boldsymbol{F}\big|_{\tan} = -\hat{\boldsymbol{n}} \times \hat{\boldsymbol{n}} \times \boldsymbol{F}$ in (2.9), where $\boldsymbol{F}$ is an arbitrary vector, one obtains the well-known EFIE for the PEC case, i.e.,

$$\boldsymbol{E}^s(\boldsymbol{J}_s)\Big|_{\tan} = -\boldsymbol{E}^i\Big|_{\tan} \tag{2.10}$$

Substituting (2.3) in (2.2) and then in (2.10), one arrives at

$$\boldsymbol{E}^s(\boldsymbol{J}_s) = j\omega\mu \iint_S \boldsymbol{J}(\boldsymbol{r}')G(\boldsymbol{r}-\boldsymbol{r}')\,\mathrm{d}S' + \frac{1}{j\omega\varepsilon}\iint_S \boldsymbol{\nabla}'\cdot\boldsymbol{J}(\boldsymbol{r}')G(\boldsymbol{r}-\boldsymbol{r}')\,\mathrm{d}S' = -\boldsymbol{E}^i \tag{2.11}$$

2.2.1.3 Discretization of EFIE

To discretize the electric field integral equation (2.11), basis functions are employed to represent the surface current.

The surface current density $\boldsymbol{J}_s$ is expanded in terms of N basis functions as

$$\boldsymbol{J}_s = \sum_{n=1}^{N} I_n \boldsymbol{f}_n(r) \tag{2.12}$$

where $\{\boldsymbol{I}_n\}_{n=1}^{N}$ are the N unknown expansion coefficients and $\{\boldsymbol{f}_n\}_{n=1}^{N}$ is the set of N basis functions.

Substituting (2.12) in (2.10) yields

$$\sum_{n=1}^{N} I_n \boldsymbol{E}^s(\boldsymbol{f}_n)\Big|_{\tan} = -\boldsymbol{E}^i\Big|_{\tan} \qquad \boldsymbol{r} \in S \tag{2.13}$$

This equation is referred to as a strong form equation, signifying that it must hold everywhere on S, which not feasible if $\boldsymbol{J}_s$ is approximated by a finite set of sub-sectional basis function. To transition from the strong form to the weak form equation, weighting functions $\boldsymbol{f}_m$, equal to the basis functions $\boldsymbol{f}_n$ are introduced by multiplying the strong form equation by a set of these test functions, so-called Galerkin's method [18], which yields

$$\sum_{n=1}^{N} I_n \Big\langle \boldsymbol{f}_m, \boldsymbol{E}^s(\boldsymbol{f}_n) \Big\rangle = -\Big\langle \boldsymbol{f}_m, \boldsymbol{E}^i \Big\rangle \tag{2.14}$$

for $m = 1, 2, \ldots, N$ and where $\Big\langle \boldsymbol{f}_m, \boldsymbol{E}^s(\boldsymbol{f}_n) \Big\rangle = \iint_S \boldsymbol{f}_m \cdot \boldsymbol{E}^s(\boldsymbol{f}_n)\, \mathrm{d}S$.

The interaction between pairs of basis functions, known as the reaction, $\Big\langle \boldsymbol{f}_m, \boldsymbol{E}^s(\boldsymbol{f}_n) \Big\rangle$, is further evaluated in terms of potentials as

$$\begin{aligned}
\Big\langle \boldsymbol{f}_m, \boldsymbol{E}^s(\boldsymbol{f}_n) \Big\rangle &= \int_V \boldsymbol{f}_m \cdot \boldsymbol{E}^s(\boldsymbol{f}_n)\, \mathrm{d}V \\
&= -j\omega \int_S \boldsymbol{f}_m \cdot \boldsymbol{A}\, \mathrm{d}S - \int_V \boldsymbol{f}_m \cdot \nabla \phi_\mathrm{e}\, \mathrm{d}V \\
&= -j\omega \int_S \boldsymbol{f}_m \cdot \boldsymbol{A}\, \mathrm{d}S - \int_S \nabla_\mathrm{s} \cdot (\boldsymbol{f}_m \phi_\mathrm{e})\, \mathrm{d}S + \int_S \phi_\mathrm{e} \nabla_\mathrm{s} \cdot \boldsymbol{f}_m\, \mathrm{d}S \\
&= -j\omega \int_S \boldsymbol{f}_m \cdot \boldsymbol{A} dS - \oint \phi_\mathrm{e} (\hat{\boldsymbol{n}} \cdot \boldsymbol{f}_m)\, \mathrm{d}\ell + \int_S \phi_\mathrm{e} \nabla_\mathrm{s} \cdot \boldsymbol{f}_m\, \mathrm{d}S \\
&= -j\omega \int_S \boldsymbol{f}_m \cdot \boldsymbol{A} dS + \int_S \phi_\mathrm{e} \nabla_\mathrm{s} \cdot \boldsymbol{f}_m\, \mathrm{d}S
\end{aligned} \tag{2.15}$$

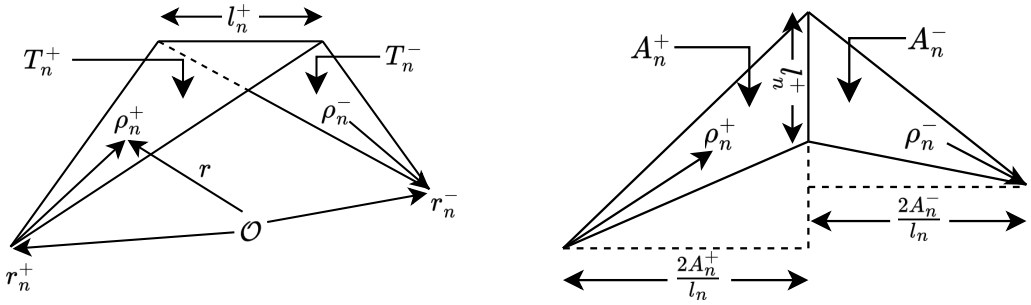
where the identity $\boldsymbol{f}_m \cdot \nabla \phi = \nabla \cdot (\boldsymbol{f}_m \phi) - \phi \nabla \cdot \boldsymbol{f}_m$ was used.

In Eq. (2.15), $\hat{\boldsymbol{n}} \cdot \boldsymbol{f}_m = 0$ is utilized, imposing a requirement on the test functions. The Rao-Wilton-Glisson (RWG) basis functions [13] are chosen to fulfill this requirement, as explained next.

Consider N RWG basis functions $\{\boldsymbol{f}_n\}_{n=1}^N$, where the n-th basis function is defined as

$$\boldsymbol{f}_n(\boldsymbol{r}') = \pm\frac{\ell_n}{2A_n^\pm}(\boldsymbol{r}' - \boldsymbol{\rho}_n^\pm) \tag{2.16}$$

over the triangles T_n^+ and T_n^- in Fig. 2.6(a). $A_n^\pm$ represents the triangle surface areas in Fig. 2.6(b) and $\boldsymbol{r}_n^\pm$ are the free corner vertices, where $\boldsymbol{\rho}_n^\pm = \pm(\boldsymbol{r} - \boldsymbol{r}_n^\pm)$.



(a) Two planar triangle, T_n^+ and T_n^-. **(b)** Surface area A_n^+ and A_n^-.

Figure 2.6: Illustration of RWG parameters on a triangle.

Let us also consider the N RWG test functions $\{\boldsymbol{f}_m\}_{m=1}^N$, i.e.,

$$\boldsymbol{f}_m(\boldsymbol{r}) = \pm\frac{\ell_m}{2A_m^\pm}(\boldsymbol{r} - \boldsymbol{p}_m^\pm) \tag{2.17}$$

The vector weighting function within the triangle at position $\boldsymbol{r}$ is defined by

$$\boldsymbol{f}_m(\boldsymbol{r}) = \begin{cases} \pm\frac{\ell_m}{2A_m^+}\boldsymbol{\rho}_m^+, & \boldsymbol{r} \in T_m^+ \\ \pm\frac{\ell_m}{2A_m^-}\boldsymbol{\rho}_m^-, & \boldsymbol{r} \in T_m^- \\ 0, & \text{otherwise} \end{cases} \tag{2.18}$$

And the surface divergence of the RWGs as

$$\nabla_s \cdot \boldsymbol{f}_m(\boldsymbol{r}) = \begin{cases} +\frac{\ell_m}{A_m^+}, & \boldsymbol{r} \in T_m^+ \\ -\frac{\ell_m}{A_m^-}, & \boldsymbol{r} \in T_m^- \\ 0, & \text{otherwise} \end{cases} \tag{2.19}$$

Inserting (2.19) and (2.18) in (2.15) yields

$$\left\langle \boldsymbol{f}_m, \boldsymbol{E}^s(\boldsymbol{f}_n) \right\rangle = j\omega\mu \int_{S_m} \boldsymbol{f}_m \cdot \int_{S_n} \boldsymbol{f}_n G dS' dS + \frac{1}{j\omega\varepsilon} \int_{S_m} [\nabla_\mathrm{s} \cdot \boldsymbol{f}_m] \int_{S_n} [\boldsymbol{\nabla}'_\mathrm{s} \cdot \boldsymbol{f}_n]\, G \,\mathrm{d}S' \,\mathrm{d}S \tag{2.20}$$

Using the function description of an RWG, Eq. (2.20) is further evaluated as

$$\left\langle \boldsymbol{f}_m, \boldsymbol{E}^s(\boldsymbol{f}_n) \right\rangle = \frac{-jk\eta\ell_m\ell_n}{4(\pm A_m^\pm)(\pm A_n^\pm)} \int_{T_m}\int_{T_n} \left[(\boldsymbol{r} - \boldsymbol{p}_m^\pm)\cdot(\boldsymbol{r}' - \boldsymbol{q}_n^\pm) - \frac{4}{k^2}\right] G(\boldsymbol{r}, \boldsymbol{r}')\,\mathrm{d}S'\,\mathrm{d}S \tag{2.21}$$

where $k = \omega\sqrt{\mu\varepsilon}$ and $\eta = \sqrt{\mu/\varepsilon}$.

If the distance between the centroids of a pair of triangles is large enough, standard Gaussian quadrature rules for triangular domains [19] are utilized. Otherwise, the singularity subtraction method is performed to handle singularities that arise in the numerical integration of Green's functions over small distances, which will be described in the next section.

2.2.1.4 Singularity Subtraction Method

The singularity subtraction method is a technique used to handle singularities that arise in numerical simulations. These singularities occur when the observation points and source points are close to each other. The singularity in the electric field integral equation is of the type $\frac{1}{R}$, as can be seen in Eq. (2.20) (Green's term). This method works by analytically subtracting the singular components of integrals, leaving only the regular components for precise numerical computation [20] . In this thesis, the Green's function is Taylor-expanded, and two odd terms are subtracted as shown by

$$G(\boldsymbol{r}, \boldsymbol{r}') = \left(\frac{e^{-jkR} - 1}{4\pi R} + \frac{k^2 R}{8\pi}\right) + \frac{1}{4\pi R} - \frac{k^2 R}{8\pi} \tag{2.22}$$

This process ensures that the integrals are regular and not singular, allowing for higher accuracy in numerical integration, while the subtracted terms can be evaluated analytically. This enhances the overall accuracy and reliability of the simulation outcomes [20].

Substituting (2.22) in (2.21) yields

$$\begin{aligned}\frac{-jk\eta\ell_m\ell_n}{4(\pm A_m^{\pm})(\pm A_n^{\pm})}\Bigg[&\int_{T_m}\int_{T_n}\left[(\boldsymbol{r} - \boldsymbol{p}_m^{\pm})\cdot(\boldsymbol{r}' - \boldsymbol{q}_n^{\pm}) - \frac{4}{k^2}\right]\left(\frac{e^{-jkR} - 1}{4\pi R} + \frac{k^2 R}{8\pi}\right)\mathrm{d}S'\,\mathrm{d}S \\ &+ \frac{1}{4\pi}\int_{T_m}(\boldsymbol{r} - \boldsymbol{p}_m^{\pm})\cdot\int_{T_n}(\boldsymbol{r}' - \boldsymbol{q}_n^{\pm})\frac{1}{R}\,\mathrm{d}S'\,\mathrm{d}S - \frac{1}{k^2\pi}\int_{T_m}\int_{T_n}\frac{1}{R}\,\mathrm{d}S'\,\mathrm{d}S \\ &- \frac{k^2}{8\pi}\int_{T_m}(\boldsymbol{r} - \boldsymbol{p}_m^{\pm})\cdot\int_{T_n}(\boldsymbol{r}' - \boldsymbol{q}_n^{\pm})R\,\mathrm{d}S'\,\mathrm{d}S + \frac{1}{2\pi}\int_{T_m}\int_{T_n}R\,\mathrm{d}S'\,\mathrm{d}S\Bigg]\end{aligned} \tag{2.23}$$

and upon introducing (From the Appendix in [20])

$$K_1^n(\boldsymbol{r}) := \int_T R^n\,\mathrm{d}S' \tag{2.24a}$$

$$\boldsymbol{K}_2^n(\boldsymbol{r}, \boldsymbol{q}) := \int_T R^n(\boldsymbol{r}' - \boldsymbol{q})\,\mathrm{d}S' \tag{2.24b}$$

the final result, using the function description of an RWG, becomes

$$\frac{-jk\eta\ell_m\ell_n}{4(\pm A_m^{\pm})(\pm A_n^{\pm})}\left[\int_{T_m}\int_{T_n}\left[(\boldsymbol{r}-\boldsymbol{p}_m^{\pm})\cdot(\boldsymbol{r}'-\boldsymbol{q}_n^{\pm})-\frac{4}{k^2}\right]\left(\frac{e^{-jkR}-1}{4\pi R}+\frac{k^2R}{8\pi}\right)\mathrm{d}S'\,\mathrm{d}S\right.$$
$$\left.+\frac{1}{4\pi}\int_{T_m}(\boldsymbol{r}-\boldsymbol{p}_m^{\pm})\cdot\left(\boldsymbol{K}_2^{-1}(\boldsymbol{r},\boldsymbol{q}_n^{\pm})-\frac{k^2}{2}\boldsymbol{K}_2^{1}(\boldsymbol{r},\boldsymbol{q}_n^{\pm})\right)-\frac{4}{k^2}K_1^{-1}(\boldsymbol{r})+2K_1^{1}(\boldsymbol{r})\,\mathrm{d}S\right] \tag{2.25}$$

where K_1^{-1}, K_1^1 and $\boldsymbol{K}_2^{-1}$ can be found in Appendix A.2.

The singularity subtraction method enhances the accuracy of the reaction between the observation point and the source points, which forms the MoM matrix. Without addressing these singularities, numerical results can become highly inaccurate.

2.2.1.5 MoM Matrix

The MoM matrix follows from (2.14), i.e.,

$$\begin{bmatrix} Z_{11} & Z_{12} & Z_{13} & \dots & Z_{1N} \\ Z_{21} & Z_{22} & Z_{23} & \dots & Z_{1N} \\ Z_{31} & Z_{32} & Z_{33} & \dots & Z_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Z_{N1} & Z_{N2} & Z_{N3} & \dots & Z_{NN} \end{bmatrix}\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_N \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_N \end{bmatrix} \tag{2.26}$$

Where $\mathbf{Z}$ is a square matrix with complex values of size $N \times N$, and both $\boldsymbol{I}$ and $\boldsymbol{V}$ are complex value matrices of size $N \times 1$.

That is,

$$\sum_{n=1}^{N} Z_{mn}I_n = V_m \tag{2.27}$$

for $m = 1, 2, \dots, N$, where $Z_{mn} = \langle \boldsymbol{f}_m, \boldsymbol{E}(\boldsymbol{f}_n)\rangle$ and $V_m = -\left\langle \boldsymbol{E}^i, \boldsymbol{f}_m \right\rangle$

Each element in the MoM matrix Z_{mn}, represents the reaction integral between the electric field radiated by $\boldsymbol{f}_n$ and tested by $\boldsymbol{f}_m$.

The matrix equations are then solved using linear algebra techniques, such as singular value decomposition, matrix inversion, or Gaussian elimination. The unknown expansion coefficients for the RWGs $\{\boldsymbol{I}_n\}_{n=1}^N$ defining the current on the geometry are determined through the inverted matrix equation

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_N \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} & Z_{13} & \dots & Z_{1N} \\ Z_{21} & Z_{22} & Z_{23} & \dots & Z_{1N} \\ Z_{31} & Z_{32} & Z_{33} & \dots & Z_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Z_{N1} & Z_{N2} & Z_{N3} & \dots & Z_{NN} \end{bmatrix}^{-1}\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_N \end{bmatrix} \tag{2.28}$$

The Method of Moments offers significant advantages when it comes to efficiently calculating the MoM matrix and manipulating it to remove the RWG basis functions. Once the MoM matrix is calculated, it allows for straightforward manipulation of this matrix to eliminate the contributions from the RWG basis functions without the need to re-calculate the MoM matrix.

2.2.1.6 Summary

The method of moments begins by taking a physical structure and applying equivalence principles to create an equivalent simplified problem [21]. The electric field integral equation (EFIE) and the magnetic field integral equation (MFIE) are derived from Maxwell's equations together with the applied boundary conditions and by introducing scalar and vector potentials to decouple the equations. The geometry is then discretized into smaller elements, such as triangles for 2D problems. Each element is associated with a set of basis functions, which are chosen based on the desired accuracy and the specific problem being solved [22].

The interaction between pairs of basis function are calculated to form a matrix known as the "MoM Matrix". This matrix describes how each basis function interacts with every other basis functions [13]. The formulated integral equations are then solved using linear algebra techniques such as singular value decomposition (SVD), matrix inversion or Gaussian elimination. The unknown currents on the geometry are determined by the solved system of equations [12].

2.3 Machine Learning

Machine Learning (ML) is a part of artificial intelligence (AI), as illustrated in Fig. 2.7, that focuses on developing statistical models and algorithms. Machine learning algorithms stand out in processing and extracting insights from large datasets. These models enable computers to execute tasks without explicit programming. Instead, these models learn from user input and output data, recognizing patterns and making decisions or predictions based on that information [23]. Machine learning allows for the automation of tasks that are time-consuming or difficult for humans to perform manually. The algorithms have the ability to detect patterns and trends in historical data that humans might overlook. This leads to more informed decisions in various domains. There are mainly two types of machine learning: unsupervised learning and supervised learning [24].

Unsupervised Learning:
Unsupervised learning involves training the algorithm on unlabeled data, where each input doesn't have a corresponding output. The algorithm learns to identify patterns or structures in the data without explicit guidance [25].

Supervised Learning:
In supervised learning, the algorithm is trained on labeled data, where each input is associated with a corresponding output. The algorithm learns from this labeled data to make decisions or predictions when encountering new data. Common supervised learning algorithms include neural networks, linear regression, decision trees, and random forests. [25].

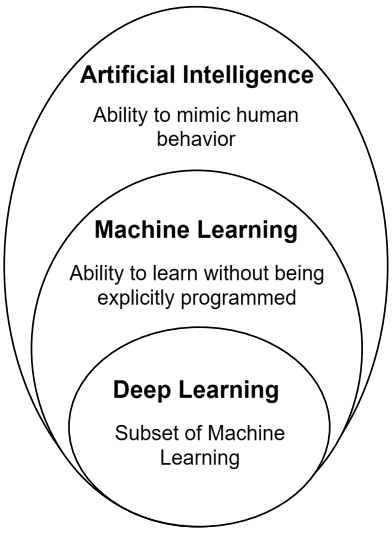


Figure 2.7: Visualization of artificial intelligence vs. machine learning vs. deep learning.

Convolutional neural network (CNN) is a specific type of deep neural network commonly used for pattern recognition tasks. Convolutional neural networks are designed to automatically and adaptively learn spatial hierarchies of features from input data using supervised learning. They have shown outstanding performance in tasks such as image classification, object detection and pattern recognition [26].

In this thesis, Convolutional neural networks are utilized, and the following sections will describe how they work.

2.3.1 Convolutional Neural Network

Convolutional neural networks are a type of deep learning model primarily used for tasks such as, image classification, object detection, and pattern recognition [26]. One of the fundamental component in the neural networks is the neuron, also known as a node or unit, illustrated in Fig. 2.8.

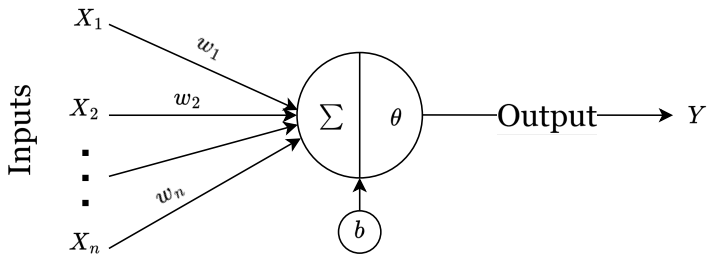


Figure 2.8: An artificial neuron.

A neuron is a basic computational unit that processes input data and passes the result to the next neurons in the network [27]. The neuron computes a weighted sum of its inputs. Mathematically, an artificial neuron is equivalent to

$$Y = \theta(\sum_{i=1}^{n} w_i X_i + b) \tag{2.29}$$

where n is the number of inputs to the neuron, X_i are the inputs, w_i are the corresponding weights, b is a bias term that allows the neuron to shift the activation function and θ is a so called activation function.

2.3.1.1 Activation Function

Activation functions introduce non-linearity into all types of neural networks, enabling them to model complex relationships in the data [28]. Common activation functions include the Rectified Linear Unit (ReLU), sigmoid and tanh. The choice of activation function depends on the task at hand: sigmoid and softmax functions are commonly used for classification problems, while ReLU is commonly used in regression tasks. The ReLU function is shown below

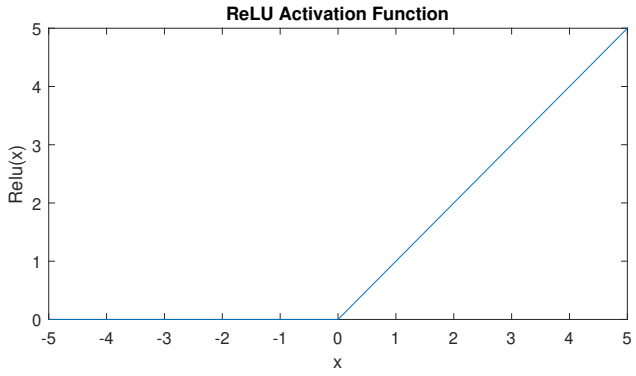


Figure 2.9: ReLU Activation Function.

and is defined as

$$\mathrm{ReLU}(x) = \begin{cases} 0, & x < 0 \\ x, & \text{otherwise} \end{cases} \tag{2.30}$$

Definitions of other activation functions can be found in Appendix A.4.

Each neuron in the neural network has an activation function, and many neurons form a layer that processes input data and transmits their output to the next layer, forming a hierarchical structure that enables the network to learn and extract complex features from the input data.

2.3.1.2 Layers

Convolutional neural networks consist of multiple layers, typically including an input layer, hidden layers, and an output layer, as illustrated in Fig. 2.10.

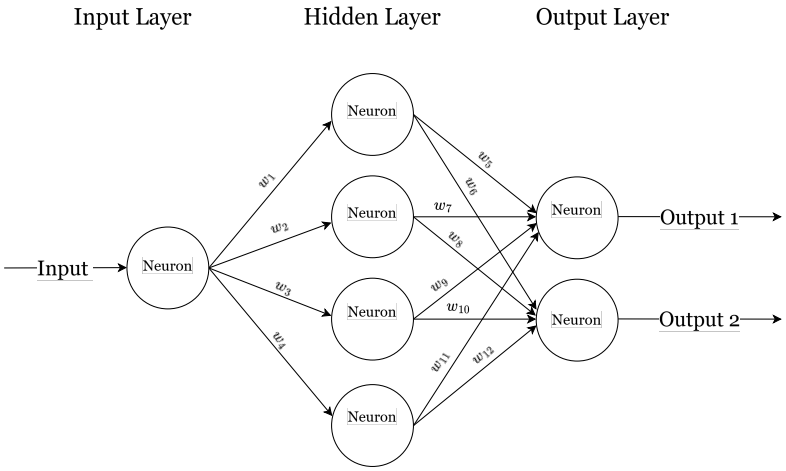


Figure 2.10: A simple neural network comprised of a input layer, a hidden layer and two output layers.

The input layer in a neural network receives raw values (features) of the input data. It consists of neurons corresponding to each input variable in the dataset. The input layer does not perform any computation. Instead, nodes in the input layer pass on the information to the hidden layers [29].

The hidden layers in a convolutional neural network consist of convolutional layers, dropout layers, batch normalization layers, and fully connected layers, which will be described in the following subsections. These layers perform computations based on the features entered through the input layer and transfer the results to the output layer. Typically, all neurons in the hidden layers use the same activation function. [29].

2.3.1.2.1 Convolutional Layer

Convolutional layers are a fundamental component of convolutional neural networks. In a convolutional layer, the input data undergoes a convolution operation. This involves sliding a set of smaller, learnable filters, called kernels, across the 2D input data with a shape of $N \times N$, as illustrated in Fig. 2.11 and Fig. 2.12.

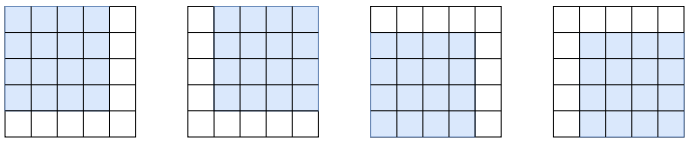


Figure 2.11: Example of a 4×4 convolutional filter on a 5×5 input data.

These filters extract specific patterns from the input data by computing the dot product between the filter and a small region of the input data at each position. [27].

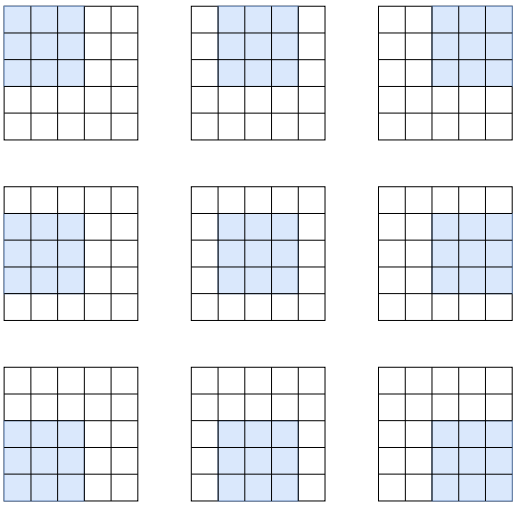


Figure 2.12: Example of a 3×3 convolutional filter on a 5×5 input data.

2.3.1.2.2 Dropout

Dropout is a regularization technique used in neural networks. The purpose of dropout is to prevent overfitting, where a machine learning model fits too closely to the training data. While this might lead to high accuracy on the training set, the model may struggle with new, unseen data. During training, dropout randomly disables a fraction of the neurons in a layer, determined by by a dropout rate. By randomly disabling neurons, it encourages the model to learn more robust features, as it cannot rely on the presence of any single neuron or combination of neurons to make predictions, thus improving the overall performance of the model on unseen data. Dropout is disabled, and the full network is used when making predictions on unseen data [27].

2.3.1.2.3 Fully Connected Layer

The fully connected layers, also known as Dense layers, are a type of neural network layer found in the hidden layers, where each neuron is connected to every neuron in the previous layer. The previous layer needs to be flattened into one dimension, using a flattening layer, before utilizing fully connected layers. Each neuron in a fully connected layer has its own weights, which are learned during the training process to capture complex relationships between features in the data. The output of a dense layer is computed using an activation function, which introduces non-linearity into the network [27].

2.3.1.2.4 Batch Normalization

A batch normalization layer is used to improve training speed and stability. It normalizes the output of a previous layer by subtracting the batch mean and dividing by the batch standard deviation. By normalizing the inputs of each layer, batch normalization reduces the internal covariate shift, which refers to changes in the distribution of layer inputs during training. This stabilization allows the network to train faster and converge more quickly [27].

2.3.1.2.5 Output Layer

The output layer is the final layer in a neural network and produces the network's prediction based on the input data. The output layer consists of one or more neurons. For classification tasks, there is typically one neuron per class, each representing the probability of that class. In a regression task, there may be one or more neurons representing the predicted values. A loss function is then used to measure the difference between the predicted output and the actual output. An optimizer is subsequently used to update the network's weights to minimize the loss function [27].

2.3.1.3 Loss Function

A loss function, also known as a cost function, is a function commonly used in a convolutional neural network. Its purpose is to measure the difference between the network's predicted output and the actual target in the training data. The loss function is utilized during the network's training process to adjust the weights and biases using optimization algorithms such as Adam. It serves as an indicator of how well the model performs on both the training data and on the validation data, and stop the training or reduce the learning rate based on validation data. The objective of training is to minimize this loss function, thereby effectively reducing the disparity between the predicted outputs and the actual targets.

For regression tasks, mean squared error (MSE) or mean absolute percentage error (MAPE) are commonly utilized.

MSE is defined as

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_{i,\text{desired}} - y_{i,\text{predicted}})^2 \tag{2.31}$$

and MAPE is defined as

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{y_{i,\text{desired}} - y_{i,\text{predicted}}}{y_{i,\text{desired}}} \times 100\% \tag{2.32}$$

2.3.1.4 Optimizer

Optimizers, such as stochastic gradient descent (SGD), Adam, and RMSprop, update the weights and biases of the convolutional neural networks during training to minimize the loss function [28]. The most recent, computationally efficient, and fastest algorithm is Adam, an adaptive moment estimator algorithm, where the learning rate is computed for each parameter [30]. This algorithm is particularly well-suited for optimization problems involving large datasets or a high number of parameters, as it imposes minimal memory requirements.

2.3.1.5 Training

Training a convolutional neural network involves several steps to optimize its parameters and enhance its predictive capabilities. In the feedforward phase, the input data traverses through the network, passing through each layer from the input layer to the output layer. At each neuron, the input data is weighted and transformed by an activation function to produce an output. The predicted output is then compared with the actual output using the loss function [28].

Following the feedforward phase, backpropagation is employed. Backpropagation is a core mechanism of neural network training, involving the updating of the network's parameters to minimize the loss function. This process computes the gradients of the loss function with respect to each parameter in the network. These gradients indicate the direction and magnitude of adjustments needed for each parameter to reduce the loss. Optimizers, such as Adam, then apply these gradients to update the weights and biases of the network [30].

The entire training process typically consists of multiple iterations, with each iteration processing one mini-batch of data. One complete pass through the entire training dataset, comprised of multiple iterations, is referred to as an epoch. During training, multiple epochs are typically performed to allow the model to learn from the entire dataset multiple times and improve its performance [27].

Evaluation of the trained neural network is crucial for assessing its performance. During training, a portion of the dataset is set aside for validation purposes. The

model's performance is evaluated on this validation dataset at regular intervals during training to monitor its progress and adjust parameters to improve performance and prevent overfitting. Once training is complete, the final trained model is evaluated on a separate test dataset to assess its ability to predict on new, unseen data. Performance metrics such as MSE and/or MAPE are computed on the test dataset to gauge the model's effectiveness in real-world scenarios. One way to improve the model's performance is to use a technique called data augmentation [29].

2.3.1.6 Data Augmentation

Data augmentation is a technique used to increase the diversity and variability of the training dataset by applying transformations such as rotation, scaling and flipping on the input data [29]. This helps to prevent overfitting and improves the robustness of the machine learning model. By exposing the model to a wider range of variations in the data, it becomes more capable of making accurate predictions on unseen data.

2.4 Genetic Algorithm

A Genetic Algorithm (GA) is a type of evolutionary algorithm, inspired by the evolutionary processes of living creatures [31]. Genetic algorithms are commonly used in search and optimization tasks where traditional methods may be ineffective. They have been extensively used in electromagnetic optimization [32] and can also be applied to machine learning models for antenna design, which will be utilized in this thesis.

Genetic algorithms operate on a population, evolving it over generations to find the best solution to a problem. The process can be described in these steps [33]:

- A population is randomly generated where each individual represents a possible solution.
- The fitness score of each individual is calculated based on a predefined fitness function, which indicates how well each solution performs with respect to the problem's objective.
- Individuals are selected from the population to serve as parents for the next generation.
- Selected individuals undergo crossover, where their genetic information is combined to produce offspring.
- After crossover, some individuals may undergo mutation, where small random changes are introduced into their genetic information. Mutation introduces diversity into the population.
- The new generation of offspring replaces the previous generation, forming the next population.

The genetic algorithm iterates through the entire process for a predefined number of generations, or until a solution satisfying a criterion is met.

3
Methods

This chapter describes the methods used in this thesis.

3.1 Mesher

Initially, a square antenna was considered due to the fact that the input shape to the convolutional neural network is of type $N \times N$ as mentioned in chapter 2.3. Within each square element in the $N \times N$ antenna, there are four triangles where each triangle can either be metal or vacuum. A script to mesh the antenna was developed within the in-house CESAR software. The script also enables users to specify all the desired input parameters, such as the length of the antenna in the x- and y-directions, the desired frequency sweep with a frequency step, the desired square size and port position where the port is the voltage-gap model from [10]. The script can also designate where to have metal or vacuum by excluding the contribution of individual triangles, each corresponding to three partially overlapping RWGs within that specific triangle. Thereafter, a full-metal antenna was meshed, as depicted in Fig. 3.1.

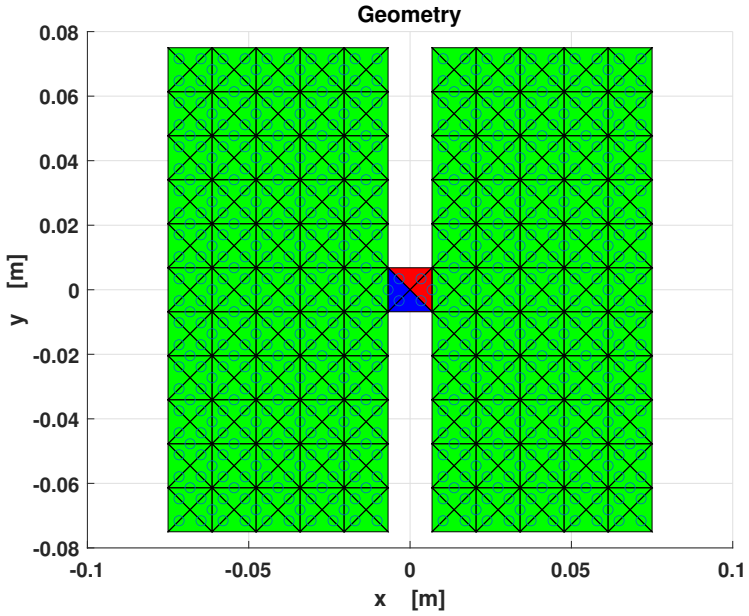


Figure 3.1: Geometry of full metal antenna where circles indicate the centers of the common edges of RWGs.

The metal component of the antenna in Fig 3.1 is modeled as a Perfect Electrical Conductor (PEC), while the empty portions are represented as vacuum, with $\varepsilon_r = 1$ and $\mu_r = 1$, and consist of an array of 11×11 squares. The antenna have the voltage-gap port in the middle and measures 15×15 cm, corresponding to $\lambda/2$ at 1 GHz. The simulation included a frequency sweep from 0.1 GHz to 5 GHz, with a frequency step of 100 MHz.

The scattering parameters and the gain of the full-metal antenna depicted in Fig. 3.1 were compared with those obtained from the commercial software CST Studio Suite 2024 to validate the in-house MoM software CAESAR. In CAESAR, two terms are subtracted from the Green's function in the electric field integral equation, and seven Gaussian quadrature points are used on the source triangle to achieve accurate results compared to CST. The results of this comparison will be presented in the next chapter.

The MoM matrix was calculated only once. During each iteration, 10–50% of the triangles were removed, resulting in the removal of the rows and columns associated with a batch of three RWG basis functions. This process effectively eliminated their influence from the matrix equation without the need to re-calculate the MoM matrix. Fig 3.2 illustrates an example of a randomly generated antenna with a portion of triangles removed.

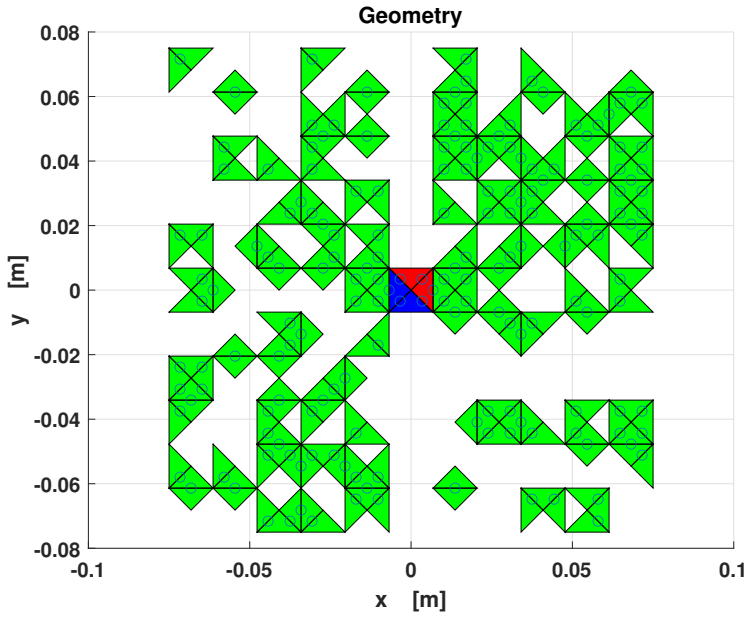


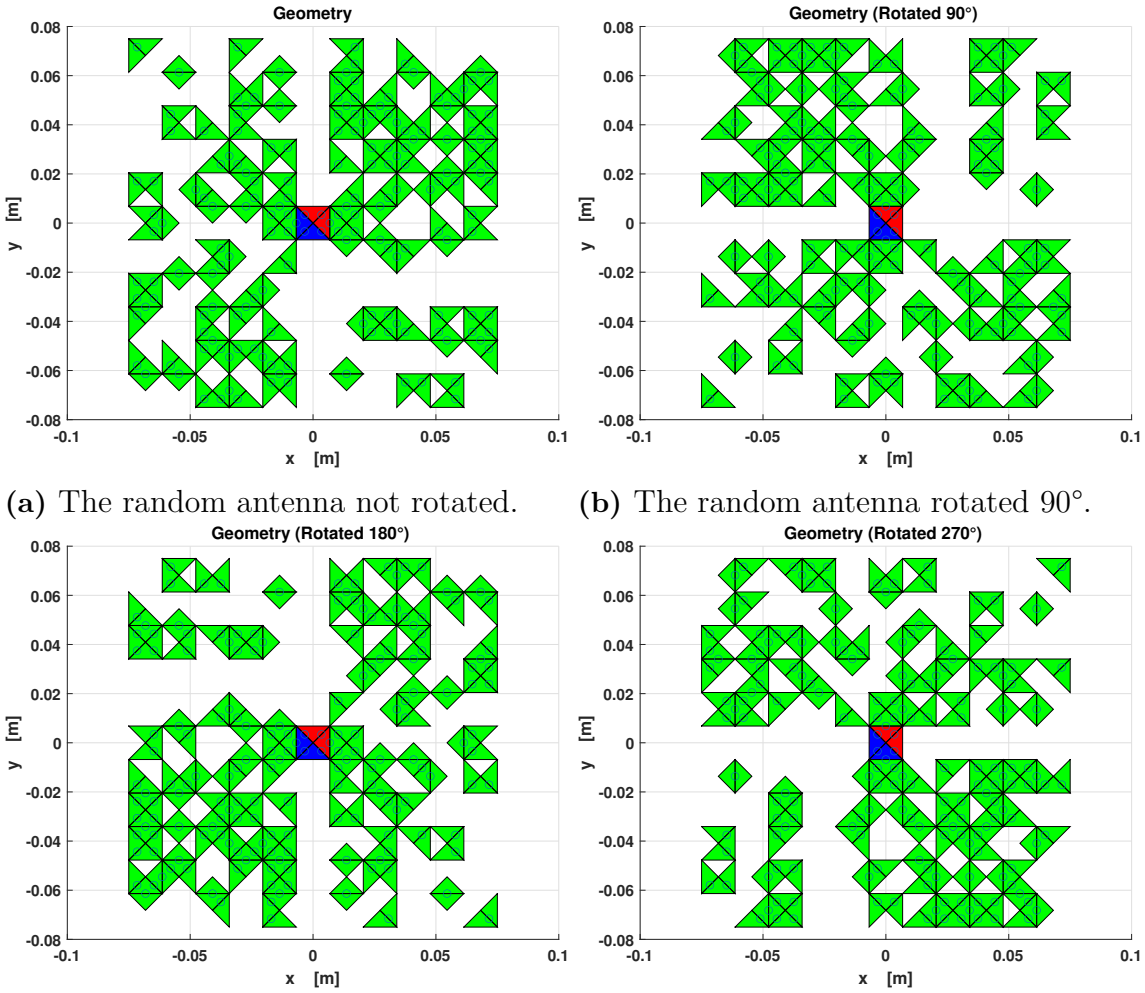
Figure 3.2: Geometry of a random generated antenna where circles correspond to RWGs.

3.2 Generation of Dataset

The dataset used in this thesis was generated on the Chalmers cluster, Vera, utilizing parallel computing on three nodes equipped with Skylake processors, each contain-

ing 32 cores. A total of 750 000 antennas were simulated, with 250 000 antennas simulated on each node. The Slurm commands used can be found in Appendix A.3.

To enhance diversity and variability, data augmentation techniques were applied to the generated dataset. This process involved rotating the antennas by 90° , 180°, and 270°, as illustrated in Fig 3.3. These augmentations aim to improve the robustness of the machine learning model. The next chapter will present the results of the generated dataset and the speed for generating it.



(a) The random antenna not rotated.

(b) The random antenna rotated 90°.

(c) The random antenna rotated 180°.

(d) The random antenna rotated 270°.

Figure 3.3: (a) shows the random antenna not rotated. (b) shows the random antenna rotated 90°. (c) shows the random antenna rotated 180° and (d) shows the random antenna rotated 270°.

The dataset was divided into three parts: one part for training, containing 70% of the dataset; another part, 15%, for validation; and the last 15% for testing (unseen data). Lastly, a comparison between the machine learning model, MoM, and CST was performed to validate the accuracy of the trained model. Metrics such as MSE, (2.31), and MAPE, (2.32), were calculated on an unseen dataset.

3.3 Machine Learning Model, Training and Optimization

A machine learning model was made to predict both the scattering parameters and the gain of the antenna. The model is structured such that each square comprises 4 channels, with each channel representing a triangle within the square. The model's parameter settings are outlined in Table 3.1.

Parameter	Settings
Optimizer	Adam
Activation Functions	Rectified Linear Unit (ReLU)
Epochs	900
Batch Size	4096
Loss function	Mean Squared Error (MAE)
Other Metrics	Mean Absolute Percentage Error (MAPE)
Dropout Rate	0.35
Filters	128

Table 3.1: Parameter settings of the machine learning model.

Fig. 3.4 illustrates the models structure, featuring convolutional layers ranging from 10×10 down to 2×2, intended to capture both large and small shapes. The ReLU activation function is applied across all layers, and a dropout layer is inserted between each convolutional layer with a dropout rate of 35%.

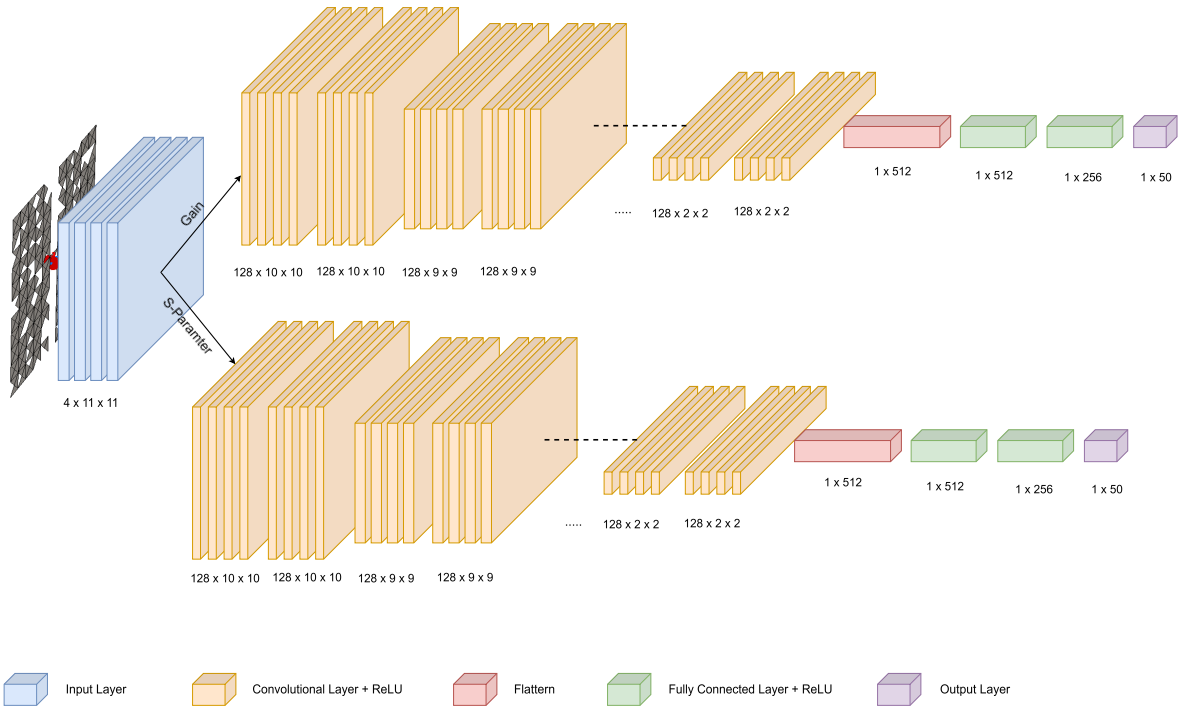


Figure 3.4: Overview of the machine learning model with scattering parameters and gain as output.

The training of the model is carried out on the Chalmers cluster, Vera, utilizing 4 A40 NVIDIA Graphics Processing Units (GPUs) to accelerate the training speed. The corresponding Slurm command can be found in Appendix A.3.

Following the model training, an optimizer utilizing the genetic algorithm is implemented. An overview of the optimization process is provided in the flowchart in Fig. 3.5. The fitness function chosen for the genetic algorithm is the mean squared error (MSE), which is used to calculate the fitness score. During the optimization process, a mask is applied to target specific S-parameters and gain profiles at desired frequencies. The frequency range of the mask is set from 0.1 GHz to 5 GHz with 100 MHz steps and a total of 40000 randomly generated antennas.

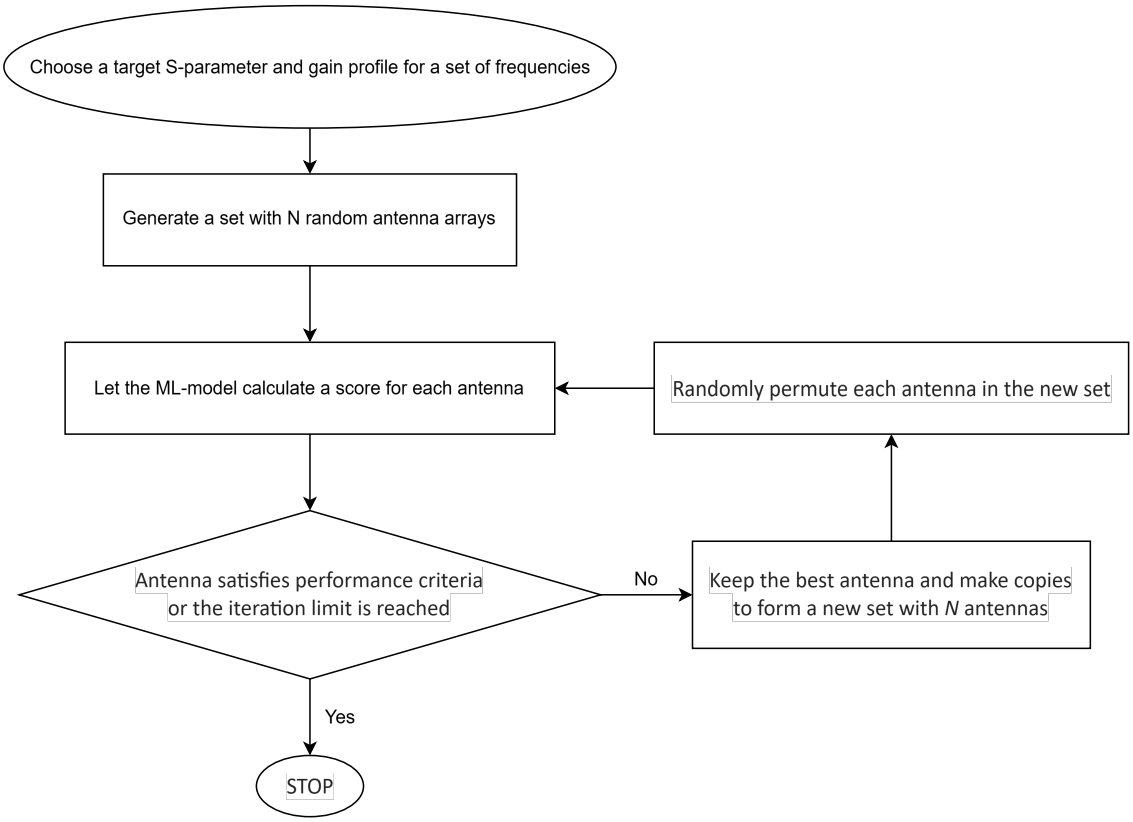


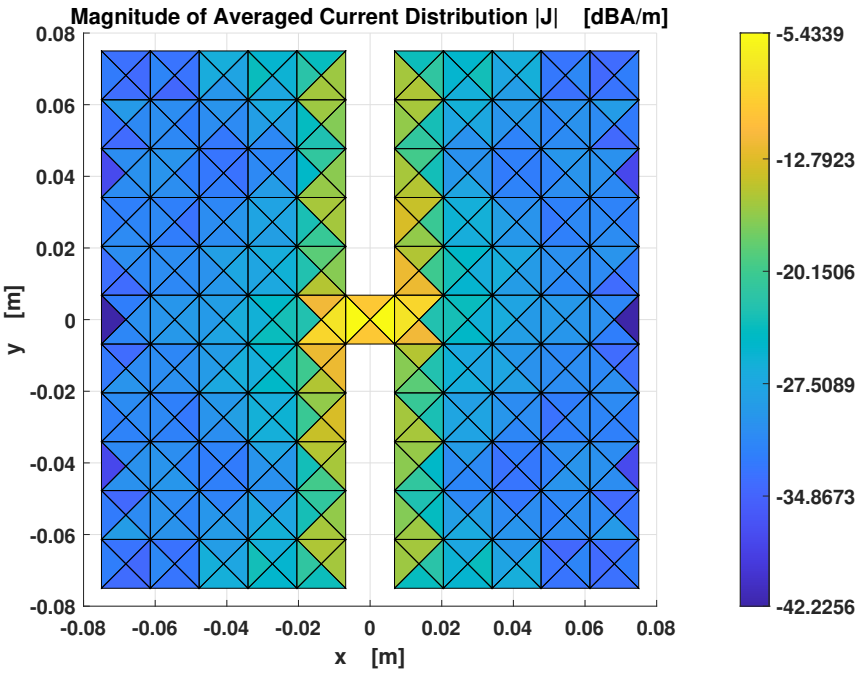
Figure 3.5: Overview of the optimization process.

4

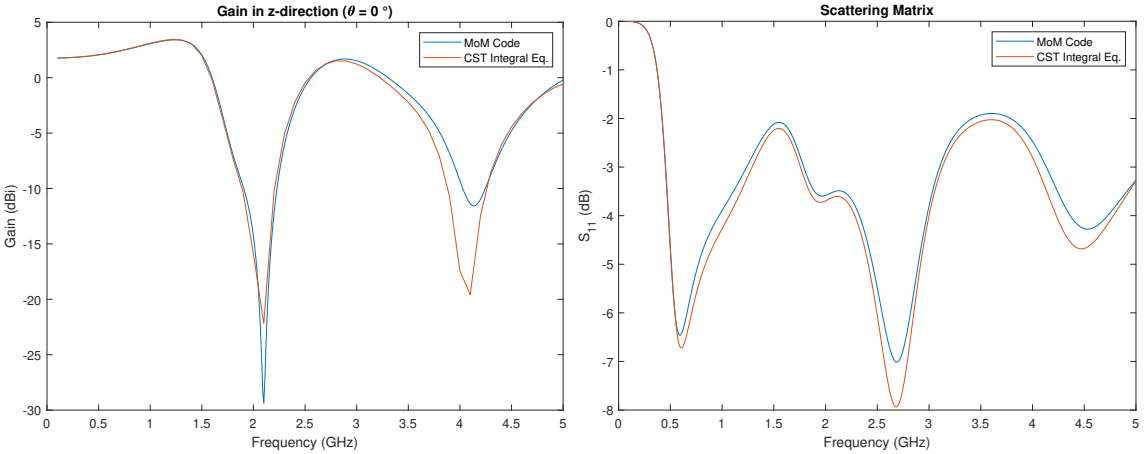
Results

In this chapter, the results from Chapter 3 are presented. The gain and scattering parameters of the full metal and the random antenna from CAESAR and CST are shown. The machine learning losses are tabulated, and the predictions for the full metal antenna and the random antenna are also presented. Lastly, the optimization of certain antennas to achieve desired scattering parameters and gain is presented.

Fig. 4.1 shows the magnitude of averaged current distribution, the gain and scattering parameters of the full-metal antenna, used to validate the CAESAR software against CST. The results show good agreement. However, the accuracy of the results is influenced by the length of the RWGs: the shorter the length, the more accurate the results. The longest lengths are vertical and horizontal, measuring $\lambda/22$ at 1 GHz and $\lambda/4.4$ at 5 GHz. It took 0.023 seconds to mesh the metal antenna using CAESAR and 1.893 seconds to simulate the scattering parameters and gain on one CPU core. In comparison, using the TCST Interface [34] for meshing the same antenna in CST took 30.012 seconds, and simulating the antenna took 10.07 seconds on the same CPU. This makes the CAESAR software more than 20 times faster for generating and simulating a full-metal antenna than CST.



(a) Current distribution of a full-metal antenna.

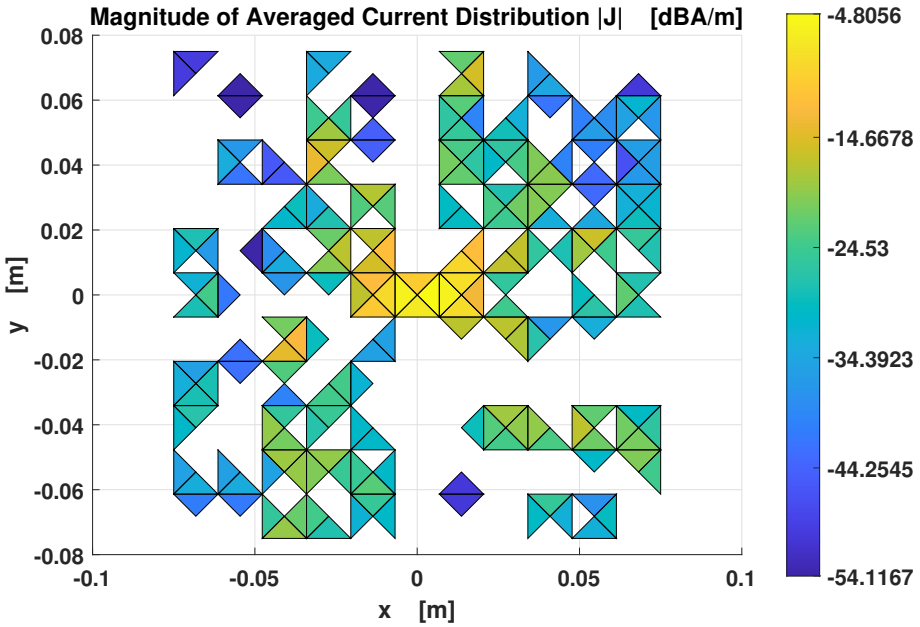


(b) Gain of a full-metal antenna.

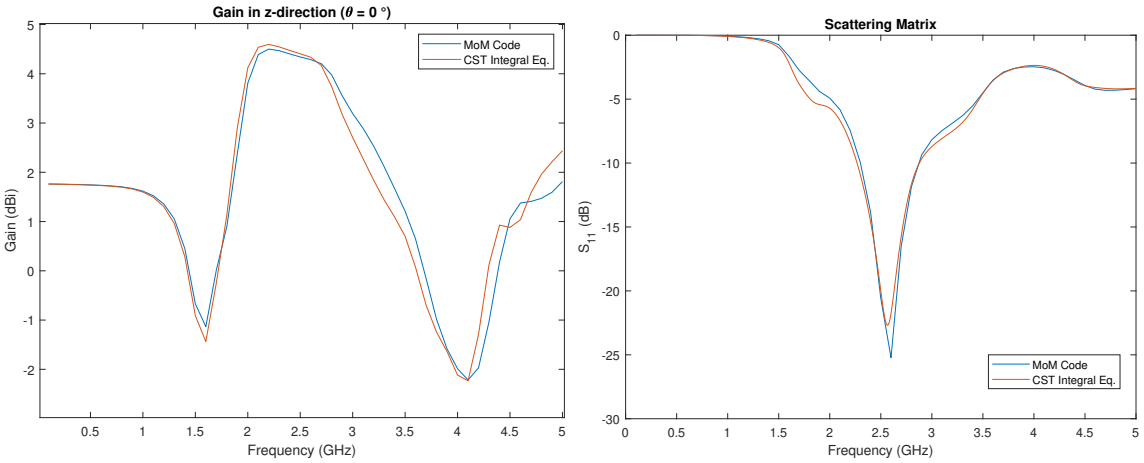
(c) S-parameter of a full-metal antenna.

Figure 4.1: Simulated results of the current distribution, gain, and scattering parameters of the full-metal antenna.

Fig. 4.2 shows the magnitude of the averaged current distribution, as well as the gain and scattering parameters of the random antenna. These results were also used to validate the CAESAR software against CST, showing good agreement and confirming the effectiveness of CAESAR. It took 0.0058 seconds to mesh the random antenna using CAESAR and 0.809 seconds to simulate the scattering parameters and gain on one CPU core. In comparison, using the TCST Interface for meshing the random antenna in CST took 10.83 seconds, and simulating the antenna took 5.31 seconds on the same CPU. This makes the CAESAR software approximately 20 times faster for generating and simulating the random antenna than CST.



(a) Current distribution of a random antenna.



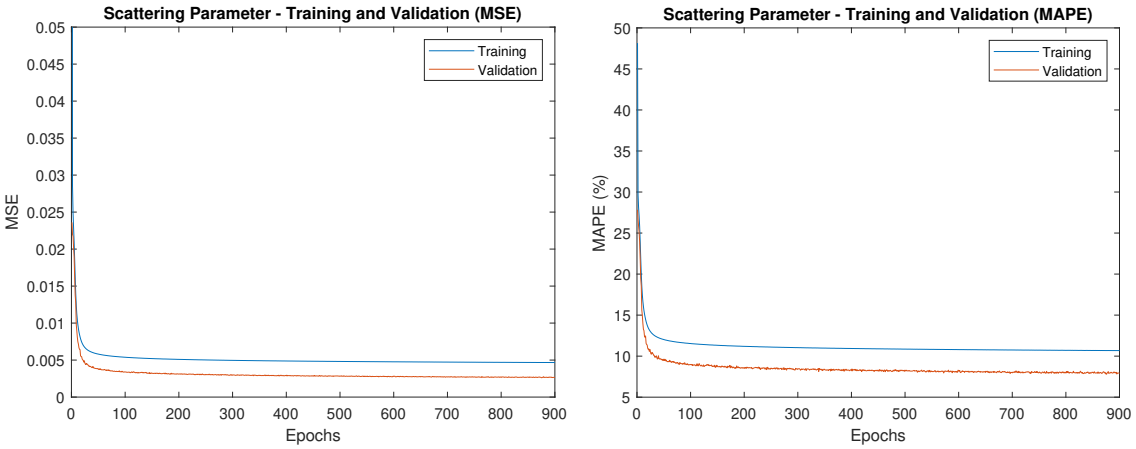
(b) Gain of a random antenna. **(c)** S-parameter of a random antenna.

Figure 4.2: Simulated results of the current distribution, gain, and scattering parameters of the random antenna.

By utilizing parallel processing on three nodes on Vera with 32 cores, the CAESAR software generated approximately 8000 antennas per minute per node. The total time for generating the dataset was 30 minutes. Through data augmentation, the size of the dataset was increased fourfold, as shown in the table below.

Dataset	Antennas
Generated Dataset	750 000
Dataset with Augmentation	3 000 000
Training Set (70%)	2 100 000
Validation Set (15%)	450 000
Test Set (15%)	450 000

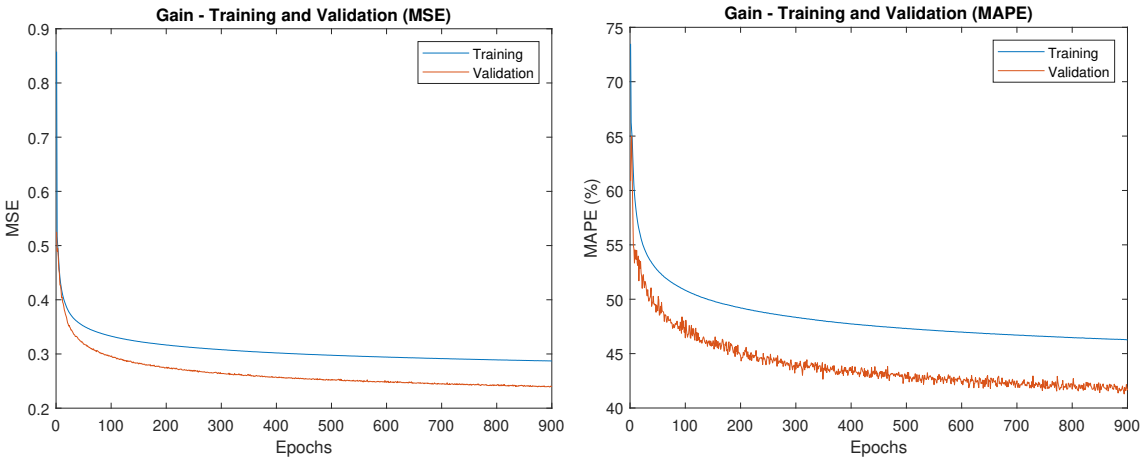
Table 4.1: Generated Dataset



(a) S-parameter MSE on unseen data. **(b)** S-parameter MAPE on unseen data.

Figure 4.3: Losses on S-parameter (a) MSE and (b) MAPE .

Fig. 4.3 and Fig. 4.4 illustrate the training and validation of the scattering parameters and gain across each epoch. Each epoch took approximately 1.5 minutes to train, and the total training time for the machine learning model was around 22 hours. However, the training could be stopped earlier as seen in the figures and still yielding accurate results.



(a) Gain MSE on unseen data. **(b)** Gain MAPE on unseen data.

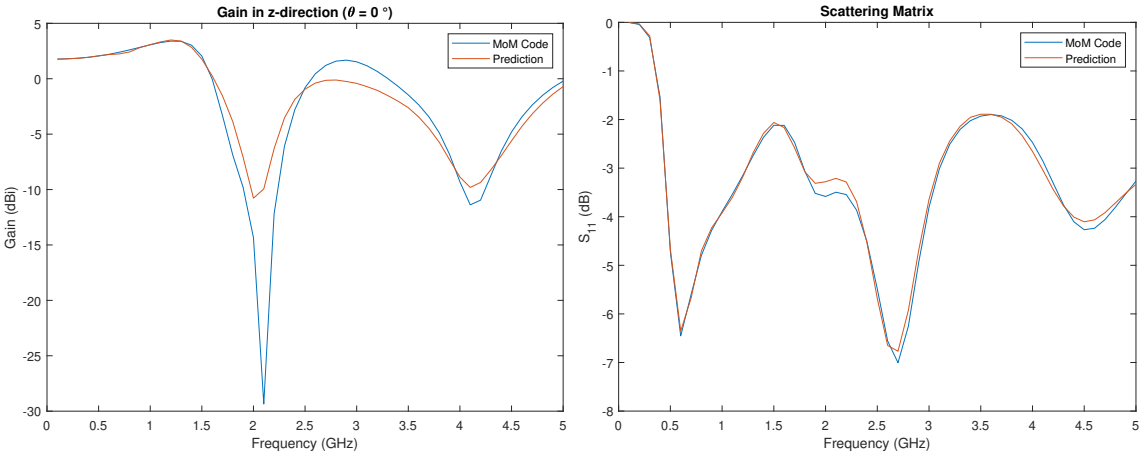
Figure 4.4: Losses on Gain (a) MSE and (b) MAPE.

Loss	Value
S-parameter (MSE)	0.0026
S-parameter (MAPE)	7.98%
Gain (MSE)	0.2369
Gain (MAPE)	41.54%

Table 4.2: Losses of the trained model on unseen dataset.

Table 4.2 presents the losses of the trained machine learning model evaluated on the 15% test dataset, which comprises unseen data not utilized during training. The loss for scattering parameters is low, while the loss for gain is high.

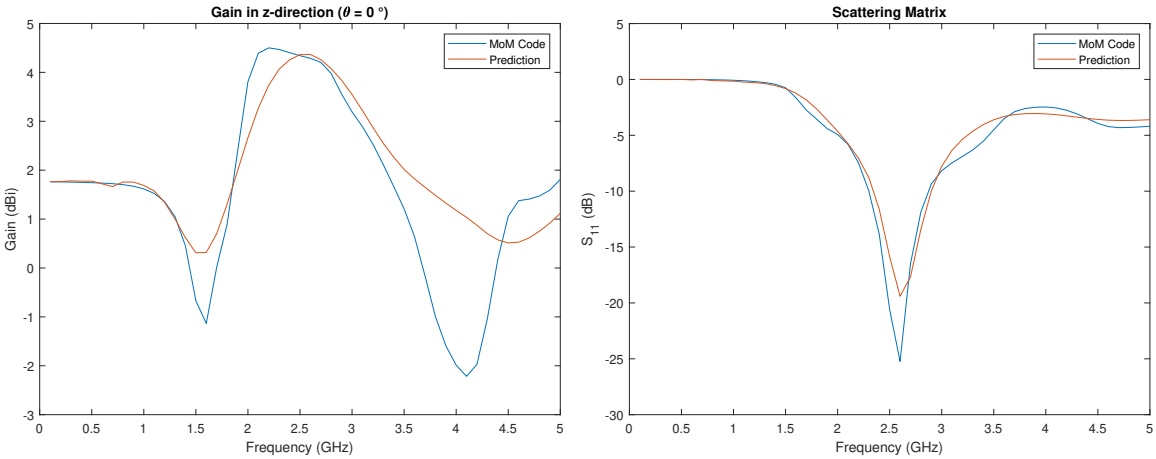
Fig. 4.5 shows the predicted gain and scattering parameters of the model for the full-metal antenna. Fig. 4.6 shows the predicted gain and scattering parameters for the random antenna, which is not included in the generated dataset. The predictions for the scattering parameters match those from the CAESAR software used to generate the dataset. However, the gain predictions, although following the correct trend, are not as accurate. This is because the gain in the z-direction is quite sensitive; even a shift of $\pm$ 10° in θ-direction would dramatically change the gain.



(a) Prediction of the gain on full-metal antenna vs MoM code.

(b) Prediction of the scattering parameters on full-metal antenna vs MoM code.

Figure 4.5: Prediction of full-metal antenna (a) Gain and (b) Scattering parameter.

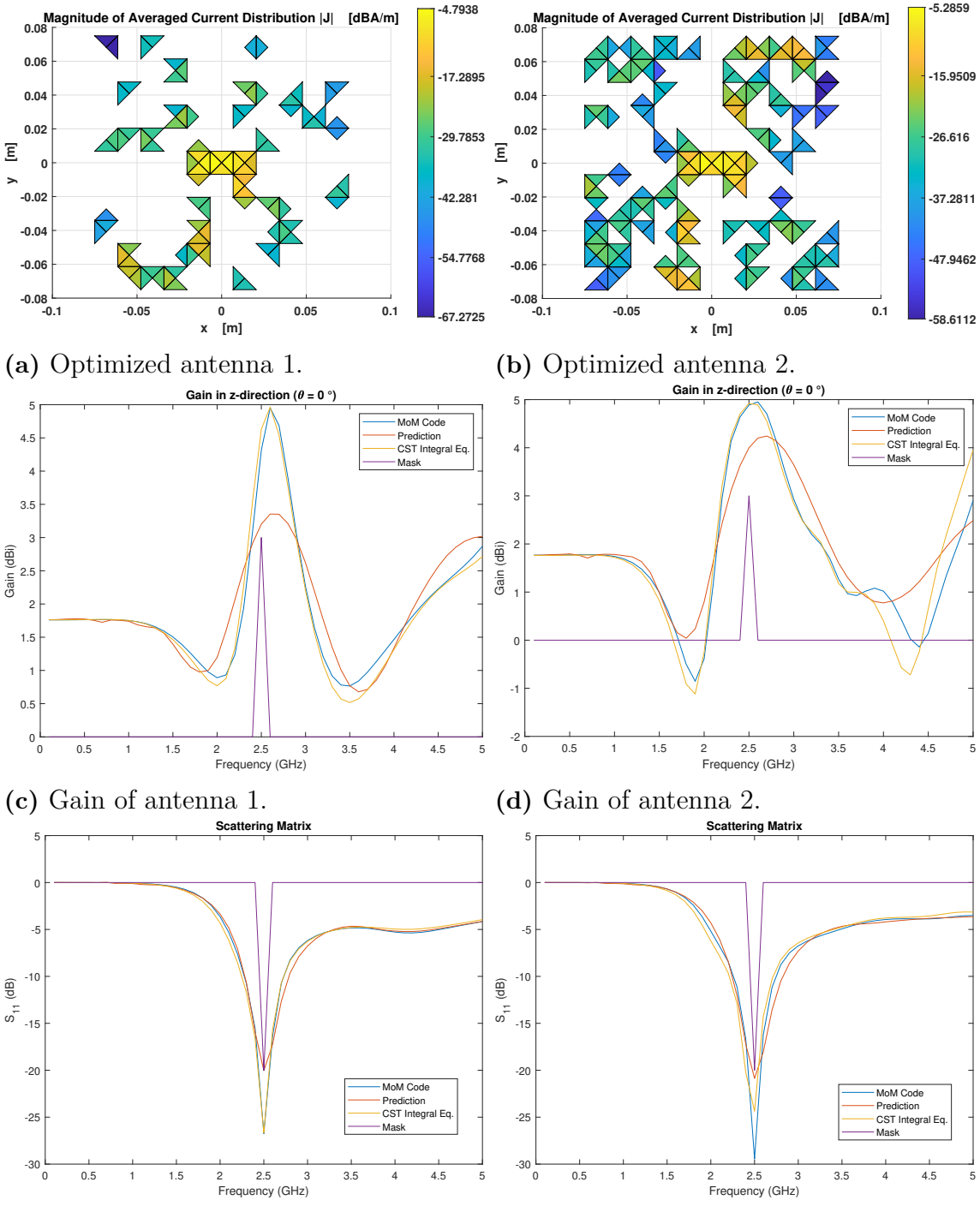


(a) Prediction of the gain on random antenna vs MoM code.

(b) Prediction of the scattering parameters on random antenna vs MoM code.

Figure 4.6: Prediction of random antenna (a) Gain and (b) Scattering parameter.

Two new optimized antennas with desired S-parameters ($S_{11} \leq$ -20 dB) and gain ($\geq$ 3 dB) at 2.5 GHz were created by the model, as shown in Fig. 4.7. The desired goals were the same as for antenna 1 and antenna 2. However, the model yielded different results in terms of the metal parts on the different antennas.



(a) Optimized antenna 1.

(b) Optimized antenna 2.

(c) Gain of antenna 1.

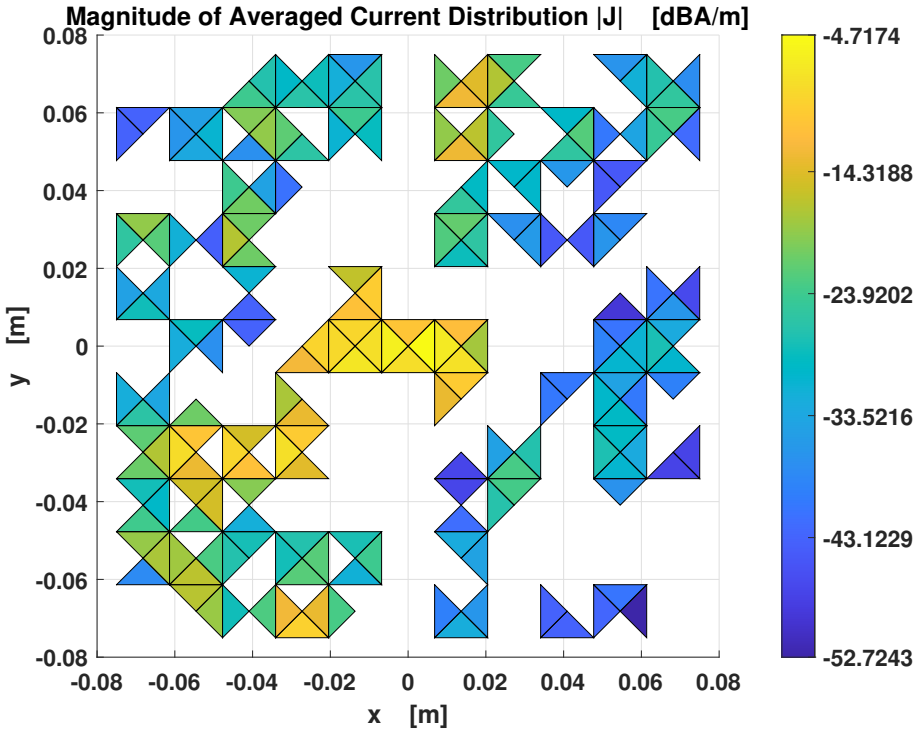
(d) Gain of antenna 2.

(e) S-parameter of antenna 1.

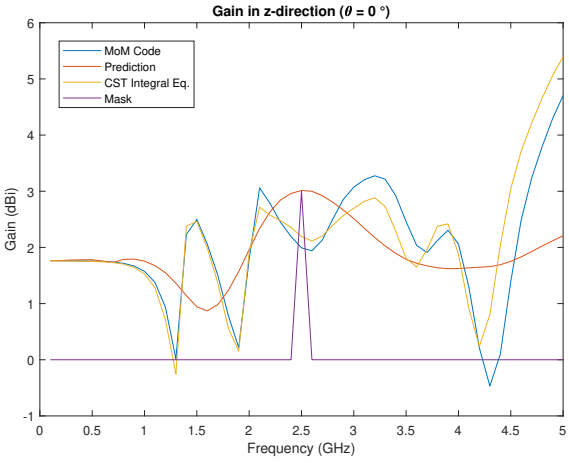
(f) S-parameter of antenna 2.

Figure 4.7: Two optimized antennas created by the model with the same goal.

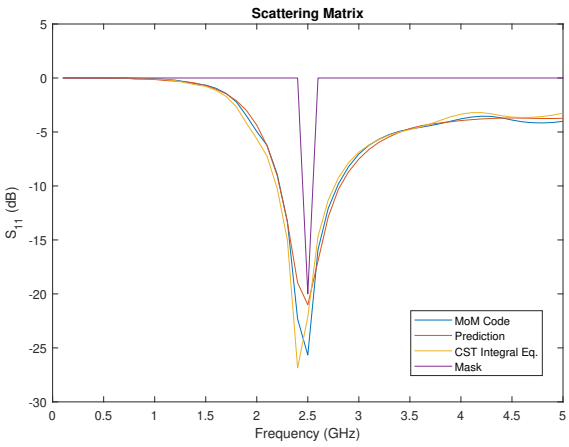
Another optimized antenna with the same desired S-parameters and gain as the previous antenna ($S_{11} \leq$ -20 dB and gain $\geq$ 3 dB at 2.5 GHz) was created by the model, as illustrated in Fig. 4.8. The scattering parameter was successfully predicted with high accuracy, but the gain prediction was less successful.



(a) Current distribution of a bad optimized antenna.



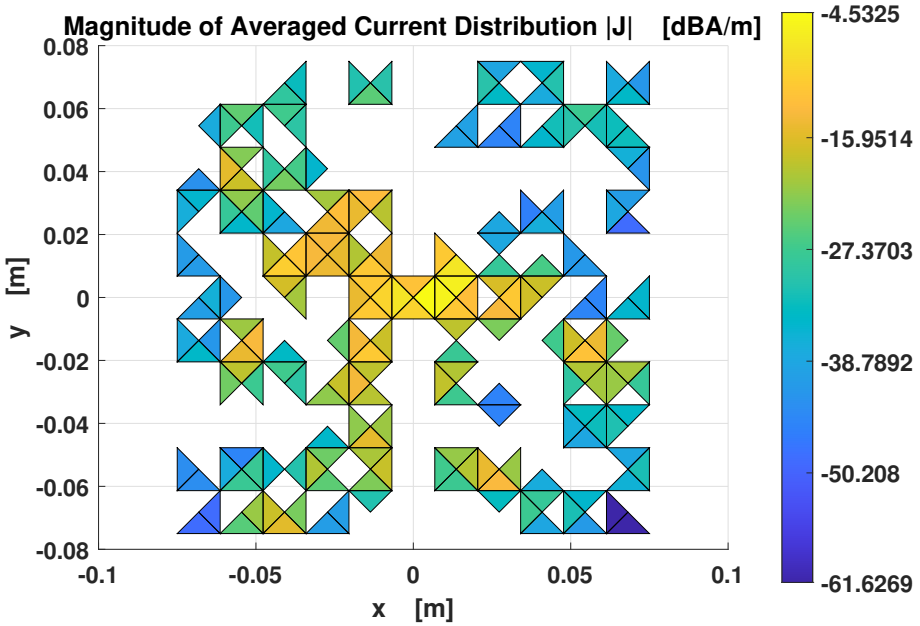
(b) Gain prediction of a bad optimized antenna.



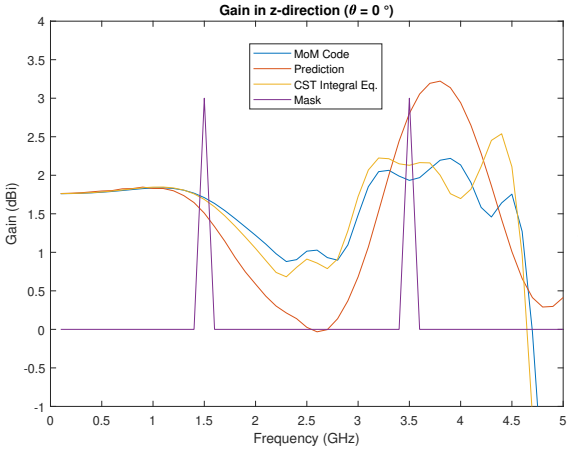
(c) S-parameter prediction of a bad optimized antenna.

Figure 4.8: One bad optimized antennas created by the model.

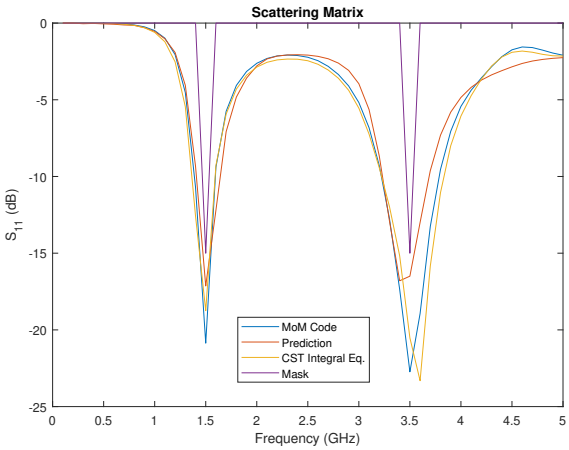
Lastly, the model optimized an antenna with resonance frequencies at 1.5 GHz and 3.5 GHz. The current distribution, gain, and scattering parameters are illustrated in Fig. 4.9, and the radiation pattern of the antenna is shown in Fig. 4.10. This demonstrates that the model is capable of predicting reasonable goals. In this particular case, the gain at 3.5 GHz was successfully achieved in the prediction, although the actual gain computed by CAESAR and CST was slightly lower. The gain at 1.5 GHz could not be achieved in the z-direction. However, Fig. 4.10 shows that the maximum gain occurs in a direction other than the z-direction and also illustrates the sensitivity of the gain in the colorbar.



(a) Current distribution of a dual resonance antenna.

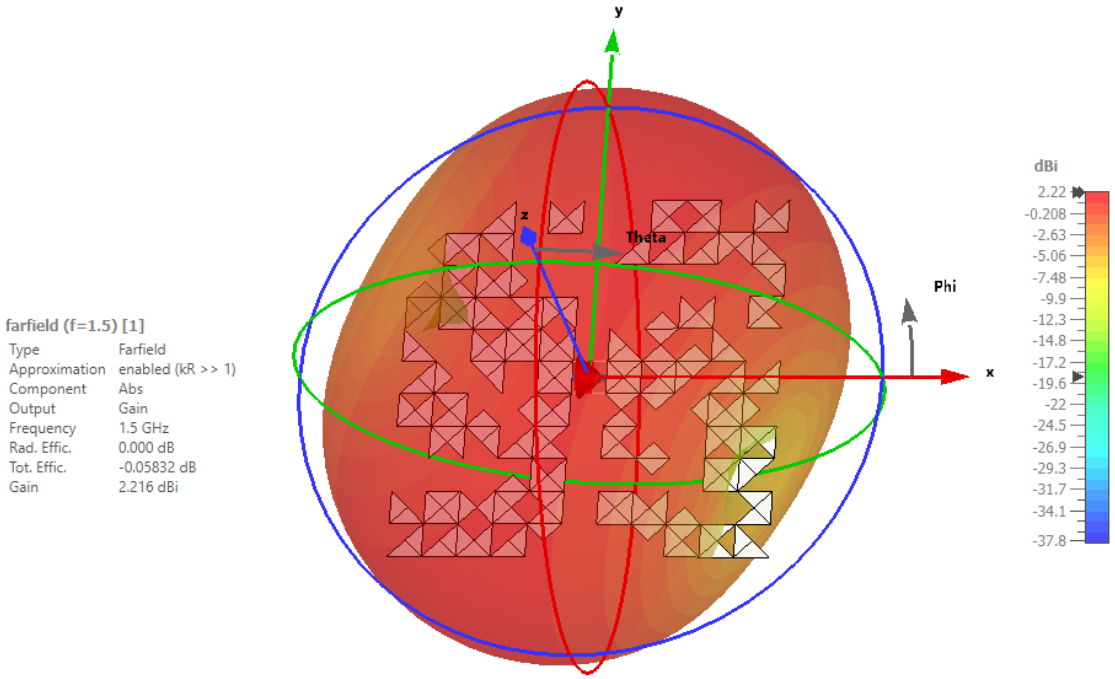


(b) Predicted gain of a dual resonance antenna.

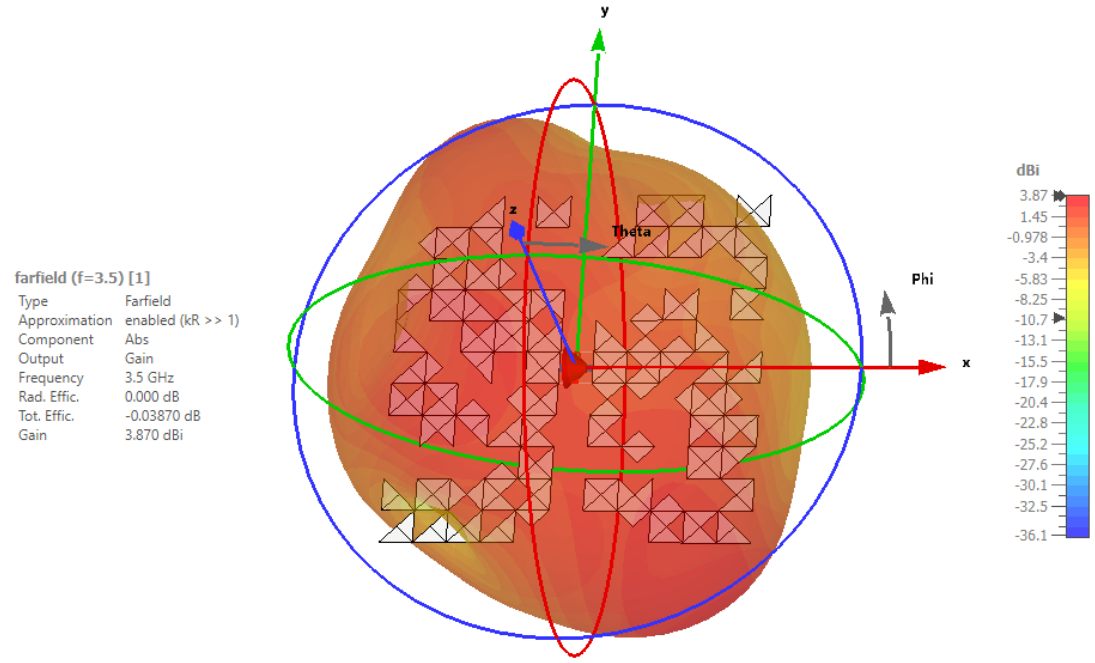


(c) Predicted S-parameter of a dual resonance antenna.

Figure 4.9: A dual resonance antenna created by the model.



(a) Radiation pattern of the dual resonance antenna at 1.5 GHz.



(b) Radiation pattern of the dual resonance antenna at 3.5 GHz.

Figure 4.10: Radiation pattern of the dual resonance antenna at 1.5 GHz and 3.5 GHz.

5
Conclusion

The aim of this thesis, which was to implement a fast Method of Moments code for efficient electromagnetic simulations and integrate a machine learning model to optimize the impedance and radiation characteristics of an antenna, was successfully achieved. This was accomplished by developing a database of simulated designs, including performance metrics, utilizing the fast Method of Moments code, and employing machine learning algorithms. The machine learning model was trained using the generated database, and it was used to optimize antenna design configurations with a genetic algorithm. Additionally, the accuracy and reliability of the machine learning models were validated through further simulations.

The predicted scattering parameters showed a high degree of accuracy, matching the simulated scattering parameters obtained using both the in-house CAESAR software and CST. However, predicting the gain with good accuracy was more challenging, as illustrated in Fig. 4.8(b) and Table 4.2, where the mean absolute percentage error (MAPE) was 41.54% and the mean squared error (MSE) was 0.2369, compared to 7.98% (MAPE) and 0.0026 (MSE) for the scattering parameters. This thesis also demonstrates the efficiency of combining machine learning with electromagnetic simulation to improve antenna design processes without the need for repeated electromagnetic simulations once the neural network model has been trained. The novelty of this work lies in the acceleration of the MoM solution, eliminating the need for re-simulation, which is traditionally a time-consuming process.

The in-house CAESAR software demonstrated remarkable efficiency, being over 2000% faster than the TCST Interface coupled with CST's integral equation solver. However, when dealing with large datasets, the bottleneck of this concept is the training speed of the machine learning model, which took approximately 22 hours, as shown in Fig. 4.4. Nevertheless, the training process could be stopped earlier, with half the time (11 hours) at 450 epochs, or less, still yielding significantly accurate results. For this antenna setup, there are approximately $2^{484} \approx 10^{145}$ possible realizations. The model will yield one solution, and different solutions satisfying the desired parameters may be given, as seen in Fig. 4.7.

The radiation pattern in Fig. 4.9 and Fig. 4.10 shows that the maximum gain of the optimized antenna could be at an angle different from the z-direction ($\theta = 0°$). In satellite communication, the gain in one direction is critical, whereas in multipath environments such as mobile phone communication, the importance of the radiation pattern diminishes due to frequent changes in the phone's orientation. Symmetry in

the antenna design ensures a predictable radiation pattern in the z-direction, which can also be achieved through the machine learning model, if desired.

5.1 Future Work

A potential extension of this work involves designing a patch antenna on dielectrics and a ground plane, modeled in accordance with the methodology presented in this thesis. This proof of concept can be further validated by manufacturing the patch antenna and conducting empirical measurements to compare with the simulated results. This would be accomplished by employing the E-PMCHWT (Poggio-Miller-Chang-Harrington-Wu-Tsai) formulation, defining also the Magnetic Field Integral Equation (MFIE), and setting the boundary conditions for the dielectric and PEC regions. The MoM solution, however, would need to be manipulated differently than by simply removing rows and columns.

To further enhance the accuracy of the model, several advanced techniques can be implemented. Increasing the order of the basis functions is one such approach, which improves the accuracy of CAESAR software and consequently enhances the model's predictions. Additionally, increasing the number of Gaussian integration points on the source and test triangles, and/or including more terms in the Green's function, can lead to more precise results. However, these adjustments typically result in only a slight increase in total simulation time, as the MoM matrix is calculated only once.

References

[1] J. H. Kim and J. Bang, "Antenna impedance matching using deep learning," *Sensors*, vol. 21, no. 20, 2021, ISSN: 1424-8220. DOI: 10.3390/s21206766.

[2] A. Aoad, "Design and manufacture of a multiband rectangular spiral-shaped microstrip antenna using em-driven and machine learning," *Elektronika ir Elektrotechnika*, vol. 27, no. 1, pp. 29–40, Feb. 2021. DOI: 10.5755/j02.eie.27583.

[3] A. F. Nazmia Kurniawati and S. Alam, "Predicting rectangular patch microstrip antenna dimension using machine learning," *Journal of Communications*, vol. 16, no. 9, pp. 394–399, Sep. 2021. DOI: 10.12720/jcm.16.9.394-399.

[4] G. A. Harris, *Reconfigurable array control via convolutional neural networks*, 2022.

[5] R. Lovato and X. Gong, "Phased antenna array beamforming using convolutional neural networks," in *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, 2019, pp. 1247–1248. DOI: 10.1109/APUSNCURSINRSM.2019.8888573.

[6] B. Sheen, J. Yang, X. Feng, and M. M. U. Chowdhury, "A deep learning based modeling of reconfigurable intelligent surface assisted wireless communications for phase shift configuration," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 262–272, 2021. DOI: 10.1109/OJCOMS.2021.3050119.

[7] J. H. Kim and S. W. Choi, "A deep learning-based approach for radiation pattern synthesis of an array antenna," *IEEE Access*, vol. 8, pp. 226 059–226 063, 2020. DOI: 10.1109/ACCESS.2020.3045464.

[8] M. D. Gregory, F. A. Namin, and D. H. Werner, "Exploiting rotational symmetry for the design of ultra-wideband planar phased array layouts," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 1, pp. 176–184, 2013. DOI: 10.1109/TAP.2012.2220107.

[9] C. X. Mao, Y. Zhang, X. Y. Zhang, P. Xiao, Y. Wang, and S. Gao, "Filtering antennas: Design methods and recent developments," *IEEE Microwave Magazine*, vol. 22, no. 11, pp. 52–63, 2021. DOI: 10.1109/MMM.2021.3102199.

[10] R. Maaskant, "Analysis of large antenna systems," English, Phd Thesis 2 (Research NOT TU/e / Graduation TU/e), Electrical Engineering, 2010, ISBN: 978-90-386-2254-5. DOI: 10.6100/IR674892.

[11] A. Taflove and S. C. Hagness, *Computational electrodynamics: the finite-difference time-domain method*, 3rd. Norwood: Artech House, 2005.

[12] C. Balanis, *Antenna Theory: Analysis and Design*. Wiley, 2015, ISBN: 9781119178989.

[13] S. Rao, D. Wilton, and A. Glisson, "Electromagnetic scattering by surfaces of arbitrary shape," *IEEE Transactions on Antennas and Propagation*, vol. 30, no. 3, pp. 409–418, 1982. DOI: 10.1109/TAP.1982.1142818.

[14] R. F. Harrington, *Field Computation by Moment Methods*. Wiley-IEEE Press, 1993, ISBN: 0780310144.

[15] C. Balanis, *Advanced Engineering Electromagnetics* (CourseSmart Series). Wiley, 2012, ISBN: 9780470589489.

[16] A. Poggio and M. Edmund, "Integral equation solutions of three-dimensional scattering problems," Dec. 1973. DOI: 10.1016/B978-0-08-016888-3.50008-8.

[17] P. S. Kildal, *Foundations of Antenna Engineering: A Unified Approach for Line-of-Sight and Multipath*. Kildal Antenn AB, 2015, ISBN: 9789163785153.

[18] W. Duncan, A. M. A. R. Committee, and G. B. A. R. Committee, *The Principles of the Galerkin Method* (ARC technical report). H.M. Stationery Office, 1938.

[19] D. A. Dunavant, "High degree efficient symmetrical gaussian quadrature rules for the triangle," *International Journal for Numerical Methods in Engineering*, vol. 21, no. 6, pp. 1129–1148, 1985. DOI: https://doi.org/10.1002/nme.1620210612.

[20] P. Yla-Oijala and M. Taskinen, "Calculation of cfie impedance matrix elements with rwg and n/spl times/rwg functions," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 8, pp. 1837–1846, 2003. DOI: 10.1109/TAP.2003.814745.

[21] W. Gibson, *The Method of Moments in Electromagnetics* (A Chapmann & Hall book). CRC Press, Taylor & Francis Group, ISBN: 9780367365066.

[22] J. Wang, *Generalized Moment Methods in Electromagnetics: Formulation and Computer Solution of Integral Equations*. Wiley, 1991, ISBN: 9780471514435.

[23] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning* (Adaptive Computation and Machine Learning), 2nd ed. Cambridge, MA: MIT Press, 2018, 504 pp., ISBN: 978-0-262-03940-6.

[24] H. M. El Misilmani, T. Naous, and S. K. Al Khatib, "A review on the design and optimization of antennas using machine learning algorithms and techniques," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 30, no. 10, e22356, 2020. DOI: https://doi.org/10.1002/mmce.22356.

[25] H. M. E. Misilmani and T. Naous, "Machine learning in antenna design: An overview on machine learning concept and algorithms," in *2019 International Conference on High Performance Computing & Simulation (HPCS)*, 2019, pp. 600–607. DOI: 10.1109/HPCS48598.2019.9188224.

[26] A. Sabharwal and B. Selman, "S. russell, p. norvig, artificial intelligence: A modern approach, third edition.," *Artif. Intell.*, vol. 175, pp. 935–937, Apr. 2011. DOI: 10.1016/j.artint.2011.01.005.

[27] M. Awad and R. Khanna, "Deep neural networks," in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Berkeley, CA: Apress, 2015, pp. 127–147, ISBN: 978-1-4302-5990-9. DOI: 10.1007/978-1-4302-5990-9_7.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, May 2015. DOI: 10.1038/nature14539.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] T. Bäck, D. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation* (Computational intelligence library v. 1). Institute of Physics Pub., 1997, ISBN: 9780750303927.

[32] D. Weile and E. Michielssen, "Genetic algorithm optimization applied to electromagnetics: A review," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 3, pp. 343–353, 1997. DOI: 10.1109/8.558650.

[33] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2, pp. 95–99, Oct. 1988, ISSN: 1573-0565. DOI: 10.1023/A:1022602019183.

[34] O. Iupikov, *TCSTInterface* (CST Studio Suite to MATLAB Interface). GitHub, 2024. [Online]. Available: https://github.com/korvin011/CSTMWS-Matlab-Interface.

[35] I. M. R. I. S. Gradshteyn and E. A. Jeffrey, *Table of Integrals, Series and Products*. MA: Academic, 1994.

A

Appendix

A.1 Gaussian Quadrature Points

Gaussian quadrature points for triangles ensures that numerical integration within triangular domains is accurate and efficient, which is essential for precise and reliable results in various computational methods and simulations. In this thesis, Gaussian quadrature points are used in chapter 3.

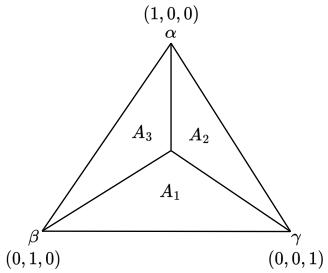


Figure A.1: Duvant triangle from [19].

Let us consider the area A in the Duvant triangle in Fig. A.1, where the co-ordinates α, β and γ are defined as

$$\alpha = \frac{A_1}{A}, \quad \beta = \frac{A_2}{A}, \quad \gamma = \frac{A_3}{A}$$

where

$$\alpha + \beta + \gamma = 1 \qquad and \qquad A = A_1 + A_2 + A_3$$

Integration is performed by a Guassian quadrature rule [19] by

$$\int_S f(\alpha, \beta, \gamma) dS = A \sum_{i=1}^{ng} w_i f(\alpha_i, \beta_i, \gamma_i) \tag{A.1}$$

where the i-th Guassian point is at $(\alpha_i, \beta_i, \gamma_i)$ in the Duvant triangle in Fig. (A.1), there is a corresponding Guassian weight w_i.

Table A.1 shows the number of Gaussian points, the weights, and the corresponding coordinates, where ng is the number of Gaussian points.

ng	weight (w)	alpha (α)	beta (β)	gamma (γ)
1	1.000000000000000	0.333333333333333	0.333333333333333	0.333333333333333
3	0.333333333333333	0.666666666666667	0.166666666666667	0.166666666666667
4	-0.562500000000000	0.333333333333333	0.333333333333333	0.333333333333333
	0.520833333333333	0.600000000000000	0.200000000000000	0.200000000000000
6	0.223381589678011	0.108103018168070	0.445948490915965	0.445948490915965
	0.109951743655322	0.816847572980459	0,091576213509771	0.091576213509771
7	0.225000000000000	0.333333333333333	0.333333333333333	0.333333333333333
	0.132394152788506	0.059715871789770	0.470142064105115	0.470142064105115
	0.125939180544827	0.797426985353087	0.101286507323456	0.101286507323456
12	0.116786275726379	0.501426509658179	0.249286745170910	0.249286745170910
	0.050844906370207	0.873821971016996	0.063089014491502	0.063089014491502
	0.082851075618374	0.053145049844817	0.310352451033784	0.636502499121399
13	-0.149570044467682	0.333333333333333	0.333333333333333	0.333333333333333
	0.175615257433208	0.479308067841920	0.260345966079040	0.260345966079040
	0.053347235608838	0.869739794195568	0.065130102902216	0.065130102902216
	0.077113760890257	0.048690315425316	0.312865496004874	0.638444188569810
16	0.144315607677787	0.333333333333333	0.333333333333333	0.333333333333333
	0.095091634267285	0.081414823414554	0,459292588292723	0.459292588292723
	0.103217370534718	0.658861384496480	0.170569307751760	0.170569307751760
	0.032458497623198	0.898905543365938	0.050547228317031	0.050547228317031
	0.027230314174435	0.008394777409958	0.263112829634638	0.728492392955404
19	0.097135796282799	0.333333333333333	0.333333333333333	0.333333333333333
	0.031334700227139	0.020634961602525	0.489682519198738	0.489682519198738
	0.077827541004774	0.125820817014127	0.437089591492937	0.437089591492937
	0.079647738927210	0.623592928761935	0.188203535619033	0.188203535619033
	0.025577675658698	0.910540973211095	0.044729513394453	0.044729513394453
	0.043283539377289	0.036838412054736	0.221962989160766	0.741198598784498
25	0.090817990382754	0.333333333333333	0.333333333333333	0.333333333333333
	0.036725957756467	0.028844733232685	0.485577633383657	0.485577633383657
	0.045321059435528	0.781036849029926	0.109481575485037	0.109481575485037
	0.072757916845420	0.141707219414880	0.307939838764121	0.550352941820999
	0.028327242531057	0.025003534762686	0.246672560639903	0.728323904597411
	0.009421666963733	0.009540815400299	0.066803251012200	0.923655933587500

Table A.1: Symmetrical Gaussian quadrature points.

If more Gaussian quadrature points are required, see [19].

A.2 Kernels

The kernels refer to the functions inside the integrals that involve the Green's function and its derivatives. These functions describe the interactions of fields from surface current. Efficient computation of these kernels, particularly when dealing with singularities, is crucial. Techniques such as the singularity subtraction method, as used in this thesis, are employed to accurately handle the singular behavior of the kernels.

In this appendix, recursive closed-form formulas are presented for the integrals [20].

$$K_1^n = \int_T R^n dS' \tag{A.2a}$$

$$\boldsymbol{K}_2^n = \int_T R^n (\boldsymbol{r}' - \boldsymbol{q})\, \mathrm{d}S' \tag{A.2b}$$

$$\boldsymbol{K}_3^n = \int_T \nabla R^n \, \mathrm{d}S', \quad \mathbf{r} \notin T \tag{A.2c}$$

$$\boldsymbol{K}_4^n = \int_T \nabla R^n \times (\boldsymbol{r}' - \boldsymbol{q}) dS', \quad \boldsymbol{r} \notin T \tag{A.2d}$$

for $n = -1, 1, 3, \ldots$ and $R = |\boldsymbol{r} - \boldsymbol{r}'|$ and $\boldsymbol{q}$ is an arbitrary vertex of a triangle T.

From [35] one gets

$$I_i^n = \int_{\partial_i T} R^n dl' = \frac{1}{n+1}(s_i^+ (R_i^+)^n - s_i^- (R_i^-)^n + n(R_i^0)^2 I_i^{n-2}) \tag{A.3}$$

where $n = -1, 3, 5, \ldots$

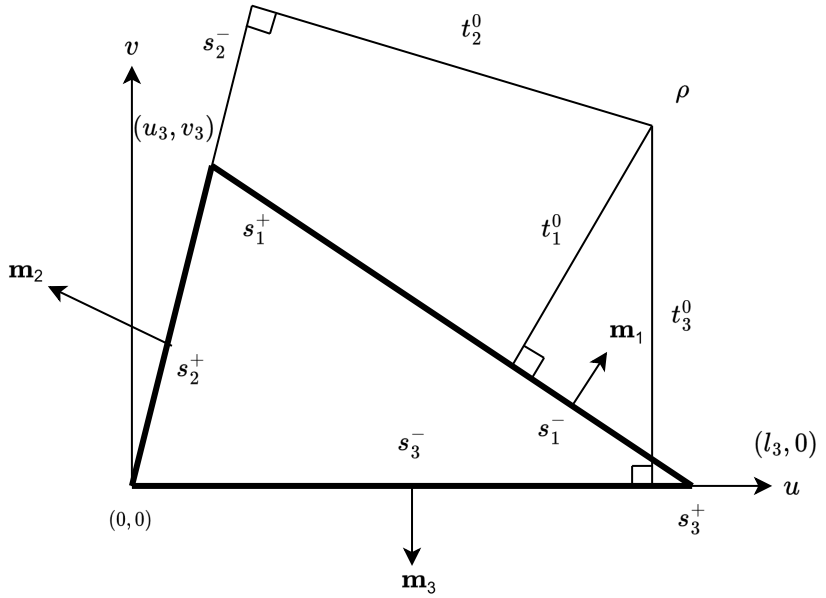


Figure A.2: Notations for analytical formulas of integrals on triangles.

$$\boldsymbol{I}_m^n = \sum_{i=1}^{3} \boldsymbol{m}_i I_i^n \tag{A.4}$$

By using (A.4) and the notations from Fig. A.2, one can write (A.2) in closed form as

$$K_1^n = \begin{cases} \frac{1}{n+2} \sum_{i=1}^{3} t_i^0 I_i^n, & w_0 = 0 \\ \frac{1}{n+2}(n w_0^2 K_1^{n-2} + \sum_{i=1}^{3} t_i^0 I_i^n), & w_0 \neq 0 \end{cases} \tag{A.5a}$$

$$\boldsymbol{K}_2^n = \frac{1}{n+2} \boldsymbol{I}_{\boldsymbol{m}}^{n+2} + (\boldsymbol{\rho} - \boldsymbol{q}) K_1^n \tag{A.5b}$$

$$\boldsymbol{K}_3^n = n w_0 \boldsymbol{n} K_1^{n-2} - \boldsymbol{I}_{\boldsymbol{m}}^n \tag{A.5c}$$

$$\boldsymbol{K}_4^n = -(\boldsymbol{r} - \boldsymbol{q}) \times \boldsymbol{K}_3^n \tag{A.5d}$$

where

$$\begin{aligned}
\boldsymbol{u} &= \frac{\boldsymbol{q}_2 - \boldsymbol{q}_1}{l_3} \\
\boldsymbol{v} &= \boldsymbol{n} \times \boldsymbol{u} \\
\boldsymbol{\rho} &= \boldsymbol{r} - w_0 \boldsymbol{n} \\
w_0 &= (\boldsymbol{r} - \boldsymbol{q}_1) \cdot \boldsymbol{n} \\
u_0 &= (\boldsymbol{r} - \boldsymbol{q}_1) \cdot \boldsymbol{u} \\
v_0 &= (\boldsymbol{r} - \boldsymbol{q}_1) \cdot \boldsymbol{v} \\
u_3 &= (\boldsymbol{q}_3 - \boldsymbol{q}_1) \cdot \frac{(\boldsymbol{q}_2 - \boldsymbol{q}_1)}{l_3} \\
v_3 &= \frac{2A}{l_3} \\
R_i^0 &= \sqrt{(t_i^0)^2 + w_0^2} \\
R_1^+ &= R_2^- = |\boldsymbol{r} - \boldsymbol{q}_3| \\
R_2^+ &= R_3^- = |\boldsymbol{r} - \boldsymbol{q}_1| \\
R_3^+ &= R_1^- = |\boldsymbol{r} - \boldsymbol{q}_2|
\end{aligned}$$

$$\begin{aligned}
s_1^- &= -\frac{(l_3 - u_0)(l_3 - u_3) + v_0 v_3}{l_1} \\
s_2^- &= -\frac{u_3(u_3 - u_0) + v_3(v_3 - v_0)}{l_2} \\
s_3^- &= -u_0 \\
s_1^+ &= s_1^- + l_1 \\
s_2^+ &= s_2^- + l_2 \\
s_3^+ &= s_3^- + l_3 \\
t_1^0 &= \frac{v_0(u_3 - l_3) + v_3(l_3 - u_0)}{l_1} \\
t_2^0 &= \frac{u_0 v_3 - v_0 u_3}{l_2} \\
t_3^0 &= v_0
\end{aligned}$$

A.3 Slurm Commands

The following Slurm commands are used to generate the dataset:

```
#!/bin/bash
#SBATCH -A project_name -p vera
#SBATCH --nodes=1
#SBATCH --cpus-per-task=32
#SBATCH --output=output_file.out
#SBATCH --error=error_file.err
#SBATCH -t d-hh:mm:ss

module purge
module load MATLAB/2023b

matlab -nojvm -nodisplay -nosplash -nodesktop -r
"matlab_name; exit;"
```

where *project_name* is the name of the project on Vera and *matlab_name* is the name of the matlab file used.

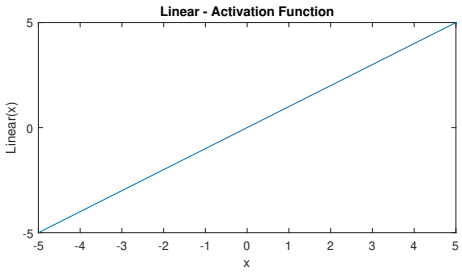
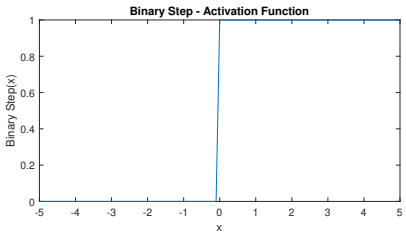
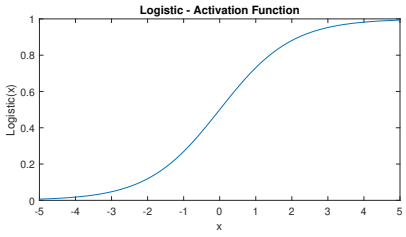
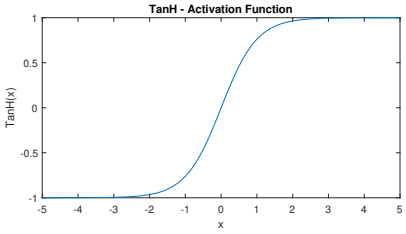
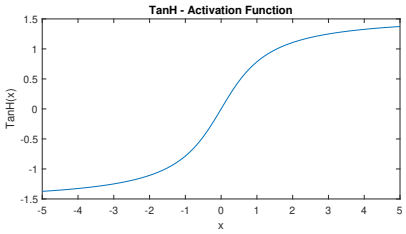
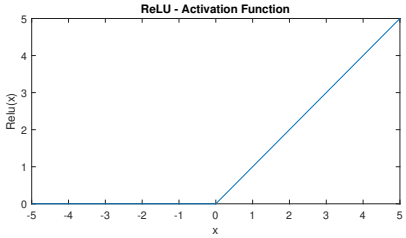
The following Slurm commands are used to train the Machine Learning model:

```
#!/bin/bash
#SBATCH -A project_name -p vera
#SBATCH --gpus-per-node=A40:4
#SBATCH --output=output_file.out
#SBATCH --error=error_file.err
#SBATCH -t d-hh:mm:ss

module purge
module load TensorFlow/2.11.0-foss-2022a-CUDA-11.7.0

python python_name.py
```

where *project_name* is the name of the project on Vera and *python_name* is the name of the Python file used. Below is a table of the available GPUs per node. The Slurm command *gpus-per-node=A40:4* can be adjusted accordingly.

#nodes	CPU	#cores	RAM (GB)	GPUs
2	Skylake	32	384	2x V100
4	Skylake	32	96	1x T4
4	Icelake	64	512	4x A40
3	Icelake	64	512	4x A100

A.4 Activation Functions

Name	Equation	Plot
Linear	$f(x) = x$	
Binary	$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases}$	
Logistics	$f(x) = \frac{1}{1 + e^{-x}}$	
tanH	$f(x) = \tanh(x)$	
arcTan	$f(x) = \tan^{-1}(x)$	
ReLU	$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases}$	

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY