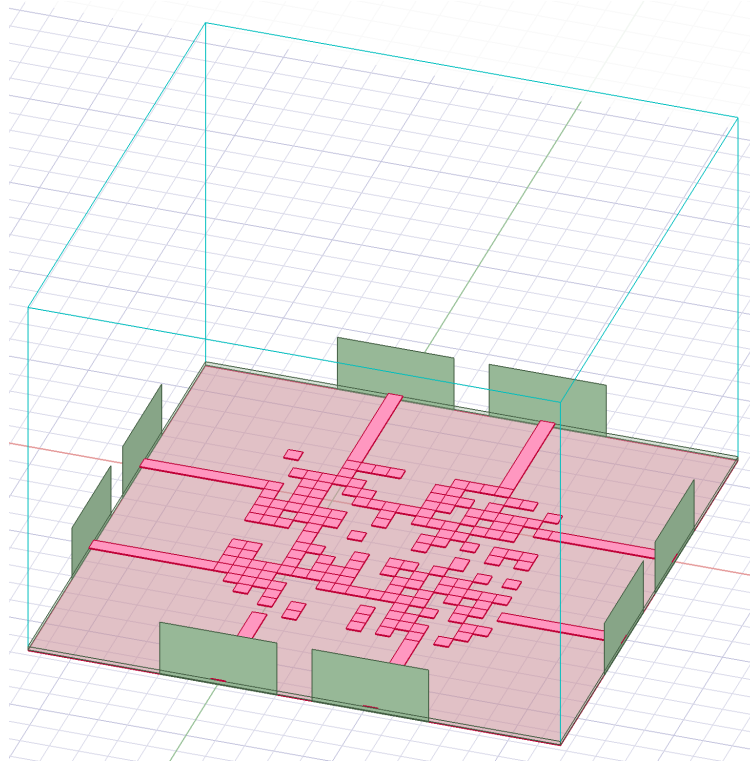




CHALMERS
UNIVERSITY OF TECHNOLOGY



Modular Machine Learning Based Circuit Design

Improving scalability of machine-learning based circuit design

Master's thesis in Systems, control and mechatronics, MSc

Jacob Ekarna, Erik Lind

DEPARTMENT OF MICROTECHNOLOGY AND NANOSCIENCE

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

MASTER'S THESIS 2024

Modular Machine Learning Based Circuit Design

Improving scalability of machine-learning based circuit design

Jacob Ekarna, Erik Lind



Department of Microtechnology and Nanoscience

MC2

Chalmers accompanied with Ericsson Research

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2024

Modular Machine Learning Based Circuit Design
Transforming traditional circuit design: Improving scalability of machine-learning
based circuit design
JACOB EKARNA
ERIK LIND

© JACOB EKARNA, ERIK LIND, 2024.

Supervisor: Dr. Martin Sjödin, Ericsson Research
Examiner: Prof. Christian Fager, MC2, Chalmers

Master's Thesis 2024
Department of Microtechnology and Nanoscience
MC2
Chalmers accompanied with Ericsson Research
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A modular circuit created by combining four circuit modules in a 2x2 configuration.

Typeset in L^AT_EX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Abstract

In traditional circuit design a pre-selected circuit topology is optimized through time consuming parameter sweeps to satisfy a design criteria. A newly introduced design concept instead utilizes machine learning models to predict the transfer function of a given circuit structure, together with a genetic algorithm to generate a circuit based on wanted scattering parameters.

The concept of machine learning in circuit design, however, has its drawbacks. One notable drawback is that the circuits generated from the model are all of the same size the model was trained on, leading to scalability issues. To overcome this problem this thesis evaluated whether or not it is possible to use a machine learning model trained on a dataset of smaller 9×9 circuits to create a larger modular circuit, consisting of four modules. The generated modular circuits were assessed by comparing the predicted scattering parameters from the optimization to the pre-selected target parameters. Additionally, simulations were performed on the generated circuits and the results were compared with the predicted parameters. The thesis also investigated if the implementation of a via fence could help isolate the modules from each other to reduce electromagnetic interference and improve performance. The differences in time efficiency between the two cases were also compared.

The results show that the modular concept works to a high degree. Based on simulation results, the root mean square error for the scattering parameters for the non-via fence model was 0.05934 and for the via fence model it was 0.04677. Adding a via fence improves the model predictions slightly and further improves the simulated circuits significantly. The results for the circuit designs with a via fence, over 100 generated circuits designs, were 13 % more accurate than the circuit designs without a via fence. However, this came at the cost of increased simulation time, as circuits using a via fence took a considerably longer time to simulate.

Keywords: Circuit design, Modular, Scattering parameters, Machine Learning, Genetic Algorithm.

Acknowledgements

We would like to gratefully acknowledge the support that we have received from our supervisor at Ericsson, Martin Sjödin. Your assistance, encouragement and constant availability during this thesis have been extraordinary, more than what we could have dreamed of. Furthermore we want to express our gratitude to additional colleagues at Ericsson who gave us a warm welcome to their department and aided us with sharp insight into softwares and ideas for our thesis. We would also like to acknowledge our examiner at Chalmers University of Technology, Professor Christian Fager. Throughout our thesis work you have been welcoming and provided directions that were essential for the thesis.

Jacob Ekarna & Erik Lind, Gothenburg, June 2024

List of Acronyms

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

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
CNN	Convolutional Neural Network
EA	Evolutionary Algorithm
ELU	Exponential Linear Unit
EM	Electromagnetic
EMI	Electromagnetic Interference
GA	Genetic Algorithm
HFSS	High-Frequency Simulation Software
I/O	Input/Output
ML	Machine Learning
NN	Neural Network
PCB	Printed Circuit Board
RF	Radio Frequency
RMSE	Root Mean Square Error
S	Scattering
Y	Admittance
Z	Impedance

Nomenclature

Below is the nomenclature of indices, sets, matrices, parameters, and variables that have been used throughout this thesis.

Indices

i, j	Indices for port number
m	Module number

Matrices

$\mathbb{S}$	Scattering matrix
$\mathbb{S}_{base}$	Base 2×2 scattering matrix
$\mathbb{Y}$	Admittance matrix
$\mathbb{Y}_{mod}$	Intermediate modular admittance matrix
$\mathbb{Z}$	Impedance matrix
$\mathbb{Z}_{int}$	Intermediate impedance matrix

Parameters

Z_0	Reference impedance
p_{metal}	Metal probability for a cell in A
p_{ports}	Port probability for a second port pair
p_{pair}	Probability of a circuit with a pathway between I/O ports in the dataset

Variables

A	Metal array matrix
N	Number of rows/columns in A
n	Number of I/O ports
$n_{connections}$	Number of interconnect ports
M	Number of Modules
D	Dimensions of $\mathbb{Y}_{mod}$
x	Input
y	Output
H	Hole/barrier diameter
R	Via fence padding diameter
d	Via fence spacing distance
s	Via fence distance to circuit
$\mathcal{N}$	Current iteration
$\mathcal{R}$	Random number

Contents

List of Acronyms	ix
Nomenclature	xi
List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 Objective	2
1.2 Research Questions	3
1.3 Disposition of Research	4
2 Theory	5
2.1 Scattering Parameters	5
2.2 Via fence	7
2.3 Machine Learning	8
2.3.1 Convolutional Neural Network	9
2.3.1.1 Convolutional layers	10
2.3.1.2 Fully connected layers	10
2.3.2 Model Evaluation	10
2.4 Genetic Algorithms	11
3 Development of Circuit Design Structure	13
3.1 Circuit structure in HFFS	13
3.2 Via Fence Distancing	14
3.3 Assembly of Modules	16
4 Development of Machine Learning Model and Genetic Algorithm	21
4.1 Data Generation and Preprocessing	21
4.2 Model Selection and Training Accuracy	23
4.2.1 Architecture selection	23
4.2.2 Evaluation of the Model	26
4.3 Genetic Algorithm	29

5	Results	31
5.1	Scattering Parameter Depiction	31
5.1.1	Full Frequency Range Analysis	31
5.1.2	Frequency Analysis	35
5.2	Time Efficiency	38
6	Conclusion	42
A	Appendix 1	I

List of Figures

1.1	Circuit design's layer structure used in HFSS simulations.	2
1.2	Illustration of the modular concept for four small-scaled circuit designs in Ansys HFSS.	3
2.1	Combination of two four-port circuits.	6
2.2	Distancing in a via fence scenario.	8
2.3	Display of layer structure of a CNN.	9
2.4	Flow chart for a genetic algorithm.	12
3.1	Circuit design's layer structure used in HFSS simulations.	13
3.2	The indexation of the HFSS model.	14
3.3	The PCB guidelines used in the thesis.	14
3.4	Illustration of the circuit used in the via fence evaluation.	15
3.5	Notation of the ports for each module and notation of the ports for the complete modular circuit.	17
3.6	Intermediate port notation during S-parameter calculation.	18
3.7	Modular circuit with unconnected ports.	20
4.1	Matrix representation of the circuit.	21
4.2	Augmentations that were conducted.	22
4.3	The training history of the baseline model.	26
4.4	The training history of the best model without via fence.	27
4.5	The training history of the best model with via fence.	28
4.6	Illustration of the genetic optimization process for a circuit consisting of 2×2 circuit modules.	30
5.1	Plot of $ S_{15} $ for prediction and target for both models.	32
5.2	Plot of $ S_{15} $ for simulation and target for both models.	32
5.3	Histograms for the RMSE scores of the predictions and simulations from both models.	34
5.4	The modular circuit used to generate the target S-parameters, as well as the two generated circuits from the GA for both cases.	35
5.5	Plot of different frequency ranges for a random example.	36
5.6	Box plots for the non-via fence model.	37
5.7	Box plots for the via fence model.	38

A.1	The result from the simulation with a cylinder diameter of 0.150 mm.	II
A.2	The result from the simulation with a cylinder diameter of 0.175 mm.	III
A.3	The result from the simulation with a cylinder diameter of 0.200 mm.	IV
A.4	Comparison of EMI on two circuits.	V
A.5	Plot of $ S_{11} $ for prediction and target for both models.	VI
A.6	Plot of $ S_{11} $ for the simulation and target for both models.	VI
A.7	Plot of $ S_{55} $ for prediction and target for both models.	VII
A.8	Plot of $ S_{55} $ for simulation and target for both models.	VII
A.9	Plot of $ S_{51} $ for prediction and target for both models.	VIII
A.10	Plot of $ S_{51} $ for simulation and target for both models.	VIII

List of Tables

3.1	Different spacings being tested for the via fence.	15
4.1	Details for the augmentations.	22
4.2	The RMSE values for the baseline model.	23
4.3	RMSE scores for different numbers of dense layers neurons.	24
4.4	RMSE scores for different numbers of dense layers.	24
4.5	RMSE scores for different number of convolutional filters.	25
4.6	RMSE scores for different number of convolutional layers and kernel sizes.	25
4.7	The structure of the baseline model.	26
4.8	The structure of the optimized non-via fence model.	27
4.9	The RMSE values for the best model for the non-via fence dataset.	27
4.10	The structure of the optimized via fence model.	28
4.11	The RMSE values for the best model for the via fence dataset.	28
4.12	Constants in the genetic algorithm.	29
5.1	RMSE for prediction and simulation from both models for different $ S_{ij} $ parameters.	33
5.2	Mean RMSE value for 100 circuits.	33
5.3	Variance and standard deviation for predictions and simulations.	34
5.4	Lists of hardware used.	39
5.5	Simulation time for each dataset.	39
5.6	The time required for training the ML model for each dataset.	40
5.7	Time for the GA optimization to reach termination requirement for each dataset.	40

1

Introduction

The development of machine learning (ML) has enabled transformative workflows in the realm of technology. As a consequence of ML, there has been a shift towards data-driven automation techniques, which help to minimize the possibilities of human error. This progress has in recent years intersected with the realm of high-frequency, electromagnetic (EM) structure design [1]. Circuits can be designed more efficiently by replacing the traditional circuit design methodologies with an automated approach using an inverse design process. An automated concept could enable optimized circuit designs that are developed in less time. Traditional circuit design is a rather time-consuming task, as a result of the multiple required parameter sweeps. The conventional approach is to manually pre-select a circuit topology which is optimized through these time-consuming parameter sweeps based on domain knowledge and design rules. Due to the limitations in domain knowledge and the increasing complexity of modern technology, constructing new, unconventional, circuit designs could be difficult. Furthermore, these parameter sweeps have to be conducted and repeated for each new circuit, leading to scalability issues.

The integration of ML offers new ways of addressing these limitations by developing an automated design process that is able to provide an optimized circuit design with desired performance. The ML-assisted approach has been covered in multiple studies [2] [3], where the main idea is to train a ML model to predict the transfer function of discretized circuit structures. These circuits are relatively simple structures solely constructed of square-shaped metal pieces [1], [2]. This enables the model to find new and improved circuit designs. Multiple studies further carry out an optimization process with the aid of a genetic algorithm (GA) [1], [4] to obtain even better results.

In order to enable the training of a ML-model used for predicting scattering parameters (S-parameters), a dataset of multiple random circuits has to be constructed. This dataset then has to be simulated to acquire each circuit's S-parameters, which in this thesis is done in Ansys High-Frequency Simulation Software (HFSS). HFSS enables an accurate simulation of the EM behaviors of different structures. Another beneficial aspect is its' compatibility with Python, thus enabling an automated scheme for creating and simulating multiple circuit structures. The circuits throughout this thesis are all constructed in a consistent manner, that is a structure where metal pieces are placed on top of a substrate. A graphic clarification of the transition between a general circuit topology to the metal plated based circuit topology can be seen in Figure 1.1. The structure of the circuit in Figure 1.1 is reflected by a $N \times N$ sized matrix (A), where each matrix element represents a metal square.

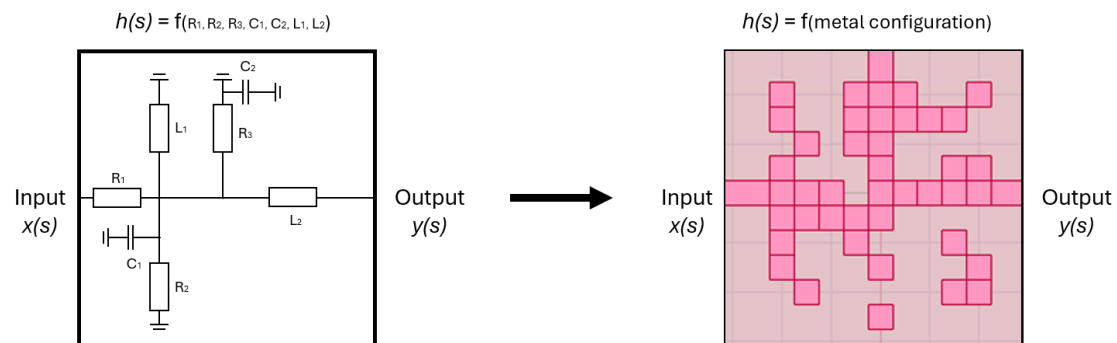


Figure 1.1: Circuit design's layer structure used in HFSS simulations.

Although the ML concept has large potential, it still has its disadvantages. Developing a well-functioning ML model could require a large amount of training data depending on the number of elements in the circuit matrices. In this case, the data is obtained through EM simulations generating the S-parameters for the circuits. As the size of the circuit structures increases, the time required to complete an EM simulation also increases significantly. On top of the increasing simulation times, the training time of the ML model also increases due to the complexity of handling more data. The possibilities to construct a model able to predict the transfer function of larger circuits is thus a time-consuming task.

1.1 Objective

The objective of this research is to examine the feasibility of connecting smaller ML-based circuits to form larger ones. This incorporates determining to what extent the transfer function, i.e., scattering parameters (S-parameters), of the interconnected circuits can be estimated from the transfer functions known from the EM-simulations. S-parameters will be presented in the frequency range of 1-10 GHz. To achieve the objective, an inverse design approach is used, instead of the traditional forward design methodologies. This entails that, after all the preparatory phases such as generating a dataset and training the ML-model, the target outcome is pre-determined while the metal pattern is decided by the model. As a result, the search for the best pattern is to be found via exploration in the design space.

The underlying idea is that the time for the simulations as well as the training of the ML-model can be decreased, due to smaller circuit sizes. Downsizing the circuits leads to reduced computational complexity as both the simulations and the ML training is performed on smaller arrays. In addition, a modular concept further alleviates the complication of scalability. This is because large circuits can be designed out of multiple smaller modules, generated by a ML-model.

The objective is mainly a proof of concept and thus the modular concept is only investigated for a smaller number of modules. All modular circuits in this work

consist of four modules, although the concept should also work for a larger number of modules. An illustration of a large circuit and the modular concept, in this case 2×2 modules, is visualized in Figure 1.2.

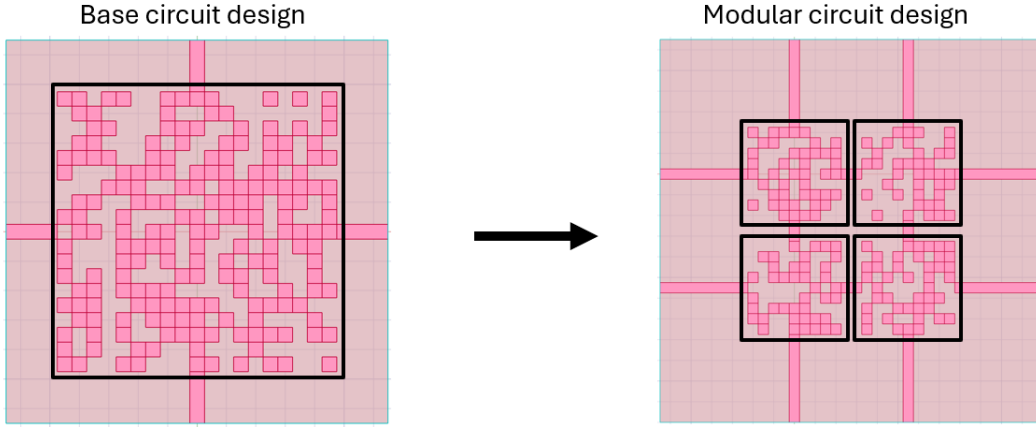


Figure 1.2: Illustration of the modular concept for four small-scaled circuit designs in Ansys HFSS.

Despite the objective focusing on the proof of concept, the thesis will cover the methodology for the complete process. Preparation of the circuit board, generation of the datasets, training of ML-models and the genetic optimization are all included in this process. Regarding the circuit structure, the objective is also to investigate the effect that electromagnetic interference (EMI) shielding structures have on the interconnected modules. These shielding structures are via fences, added around the pixel array to isolate each module. The main idea is to analyze whether the addition of via fences leads to a better performance compared to a circuit without them.

1.2 Research Questions

The objective of the study is conducted by bringing forth sufficient answers to the following research questions:

- To what extent can the modular concept represent the S-parameters? Is the representation better for certain frequencies in the 1-10 GHz span?
- To what extent does EMI shielding structures impact the S-parameter accuracy of the generated circuits? Does the representation of the S-parameters vary compared to the non-EMI shielding design?
- What affects the time consumption for the modular concept? Does the time consumption change due to the addition of a via fence?

1.3 Disposition of Research

This research is organized into four different sections. In the upcoming three sections, i.e., *2 Theory*, *3 Development of Circuit Design Structure* and *4 Development of Machine Learning Model and Genetic Algorithm*, the focus is set on the methodology used to obtain the results. Section *2 Theory* covers the background knowledge regarding ML and GA concepts which are used in this research. This section also covers the basics of a via fence and scattering parameters. Sections *3 Development of Circuit Design Structure* and *4 Development of Machine Learning Model and Genetic Algorithm* mainly focuses on the implementation of everything mentioned in Section *2 Theory*. The subsections within Section *3 Development of Circuit Design Structure* and *4 Development of Machine Learning Model and Genetic Algorithm* documents the outcomes of different via fence spacing as well as performance of different ML-model structures. Ultimately, section *5 Results* and *6 Conclusion* answers the research questions and bring forward relevant conclusions regarding the modular concept.

2

Theory

To encapsulate the complete development of the ML-based concept, the theory section has been divided into four main parts. The first two encompass the theory regarding elements in general circuit design. The latter sections discuss concepts within the realm of ML and GA.

2.1 Scattering Parameters

Scattering parameters are used to describe and characterize the behaviour, i.e., the transmission and reflection, of an electrical network [5]. Essentially, they interpret how an incident wave is transferred between ports. The S-parameters of a system is usually displayed in matrix format, where the number of rows and columns are dependent on the number of input/output (I/O) ports. In the basic case with a two port system, there are four S-parameters as seen in Equation (2.1),

$$\mathbb{S}_{base} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \quad (2.1)$$

where S_{11} and S_{22} describe the reflection of the input signal at each port. S_{12} and S_{21} however describe the transmission of the signal [5].

The S-matrix ($\mathbb{S}$) is a $n \times n$ matrix and thus consists of n^2 elements. The elements in $\mathbb{S}$ describe the magnitude and phase of the signals. Thus, the transmission from port 1 to port 2, and from port 2 to port 1 will always be the same, i.e., $S_{12} = S_{21}$. This is not true for the reflection elements, as long as each port is connected to any transmission line, i.e., not unconnected. In that case, there will be, losses aside, full reflection from each port.

Observing the S-parameters is important for networks at radio frequency (RF) and microwave frequencies [6]. This is due to the fact that S-parameters can adequately describe the system over multiple frequencies. In comparison, the same analysis is more complicated to perform with the admittance and/or impedance representation. However, these complementary representations of a circuit's behaviour, i.e., impedance parameters (Z-parameters) and admittance parameters (Y-parameters), are still useful in calculations of steady-state S-parameters. Z-parameters are used to quantify the impedance between ports, whilst the Y-parameters describe the admittance. Both the Z-parameters and the Y-parameters are represented by matrices with the same structure as $\mathbb{S}$. The conversion between the three parameters is essential for being able to calculate the S-parameters of interconnected modules. When

combining two modules, a port from one circuit has to be connected to with a port from the other circuit. Thus, the size of the each parameter matrix will be changed to reflect the number of ports in the connected circuit. An example of this is illustrated in Figure 2.1 where two four-port circuits have been combined and the total number of ports have gone from eight to six.

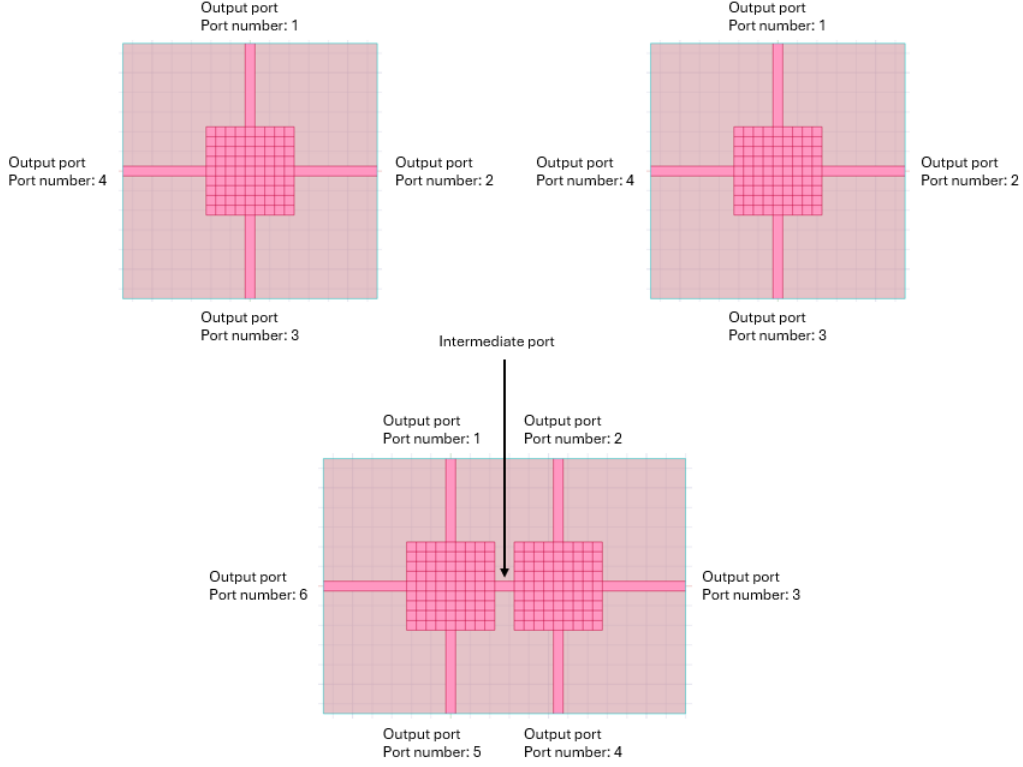


Figure 2.1: Combination of two four-port circuits.

The resulting $\mathbb{S}$ for this combination is found by first converting it to its representative admittance form, i.e., a Y-matrix ($\mathbb{Y}$). Converting between Y-parameters and S-parameters is in this case done due to the simple nature of addition of Y-parameters. The conversion between $\mathbb{S}$ and $\mathbb{Y}$ can be done as shown in Equation (2.2) [7],

$$\mathbb{Y} = \frac{1}{Z_0}(\mathbb{I} + \mathbb{S})^{-1}(\mathbb{I} - \mathbb{S}) \quad (2.2)$$

where Z_0 is the reference impedance, which is the general impedance for the transmission line. In this research, and in circuit design in general, Z_0 is set to $50 \, \Omega$. $\mathbb{I}$ is an identity matrix with the same size as $\mathbb{S}$.

As a result from the conversion in Equation (2.2), each module has its own respective $\mathbb{Y}$. To proceed with the combination of modules, the multiple modules' $\mathbb{Y}$ has to be inserted into a larger modular Y-matrix ($\mathbb{Y}_{mod}$). This new matrix has the dimensions (D), rows and columns, calculated as shown in Equation (2.3),

$$D = (M \times n) - n_{connections} \quad (2.3)$$

where M is the number of modules, n is the number of ports per module and $n_{connections}$ is the number of connected ports between modules. The number of connected ports represents the number of connections between different modules. In Figure 2.1 the intermediate port for the modular circuit is a connection between port one and port four in the respective modules.

In the example in Figure 2.1, the variable values are $N = 2, n = 4, n_{connections} = 1$, resulting in $D = 7$. $\mathbb{Y}_{mod}$ can thus be filled in place by each module's $\mathbb{Y}$. For the example in 2.1, the new port number for the modular circuit determines at which row and column the respective Y-parameter is inserted into $\mathbb{Y}_{mod}$. In the case of the intermediate port, the Y-parameter from each module is added. This will be further illustrated in Section 3. Before reverting back to the $\mathbb{S}$, the unused I/O-ports must be handled either by disconnecting the ports or short circuiting them. Managing the unused I/O-ports is easier to work with by first converting the Y-parameters to Z-parameters. Thus, $\mathbb{Y}_{mod}$ can be inverted to an intermediate Z-matrix ($\mathbb{Z}_{int}$) as displayed in Equation (2.4).

$$\mathbb{Z}_{int} = \mathbb{Y}_{mod}^{-1} \quad (2.4)$$

The resulting $\mathbb{Z}_{int}$ will have the same dimensions as $\mathbb{Y}_{mod}$. However, when the unused ports have been eliminated, the number of S-parameters left is n^2 . The resulting Z-matrix $\mathbb{Z}$ will then have the same size as the intended final $\mathbb{S}$. As a final step, Equation (2.5) is used for the reversion back into the new $\mathbb{S}$ for the interconnected modules.

$$\mathbb{S} = (\mathbb{Z} + \mathbb{I} \cdot Z_0)^{-1}(\mathbb{Z} - \mathbb{I} \cdot Z_0) \quad (2.5)$$

This calculation will have to be performed individually for all chosen frequency points in the frequency span.

2.2 Via fence

A via fence is a structure commonly found on printed circuit boards (PCBs), due to its capability of mitigating EMI between components. This is used to ensure that each part on the circuit board performs as intended without outside disturbance or crosstalk [8]. The via fence functions by shielding the wave in the substrate in the same way it is shielded in downward by the ground plane. The via fence is, generally, structured in even spaced rows between different sections. It consists of a cylindrical structure, either holes or small barriers, that form the fence. A visualization of a via fence can be seen in Figure 2.2.

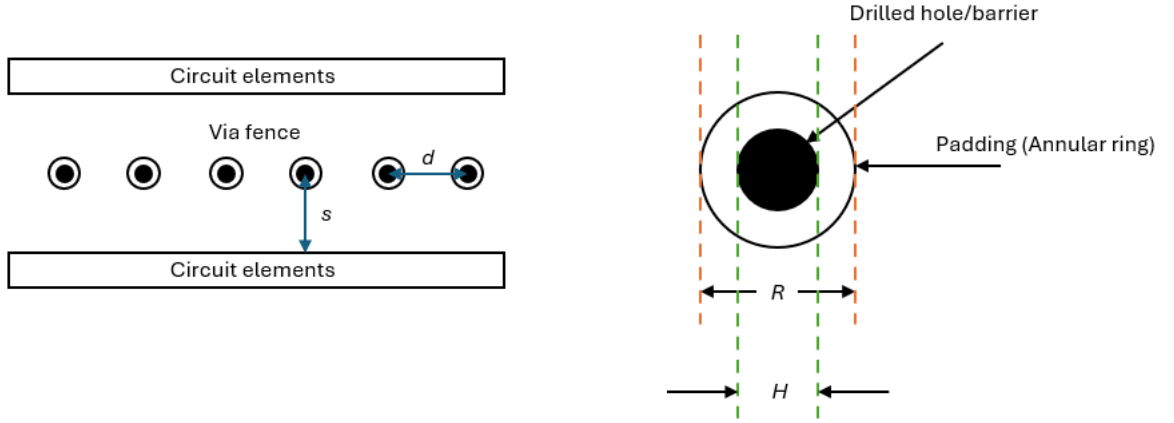


Figure 2.2: Distancing in a via fence scenario.

Design configurations are rather important to analyze when designing a via fence. The via fence must be within the design boundaries set by PCB manufacturers. In Figure 2.2 shows important distances to have in mind. In the magnified illustration of a via fence cylinder, seen in Figure 2.2, each cylinder is constructed by two parts: a hole/barrier and padding around the hole/barrier. Diameter for the hole/barrier H and diameter of the padding R are essential to have in mind when selecting the spacing distance d . The distance d refers to the distance between the centers of two different fence cylinder, and thus it must follow the condition shown in Equation (2.6).

$$R < d \quad (2.6)$$

Furthermore, when creating the via fence the distance between the circuit elements and the via fence s must be according to Equation (2.7).

$$\frac{R}{2} < s \quad (2.7)$$

If the via fence cylinders' padding were to come into contact, this could lead to short circuits as it creates an unwanted electrical connection between the layers in the PCB.

2.3 Machine Learning

Machine learning is a data focused subcategory within the field of artificial intelligence (AI). The central principle is to develop algorithm-based models, trained using large amount of data, that can then make predictions on new data. This is achieved through iterating over all data examples to find underlying patterns between input and output. A trained model is then, to a certain extent, able to make a prediction or classification based on the input data [9].

There are three general ML subcategories to choose from which dictate how the model is created. These are unsupervised learning, supervised learning and reinforcement learning (not to be discussed in this thesis), where each category is favorable for different tasks. The appropriate category can be chosen depending on the type of data used. Unsupervised learning refers to an algorithm which is provided with multiple data points as input and learns to find distinguishing traits [10]. It thus attempts to find an optimal way of separating the training data by looking for unique features. This translates well to finding hidden patterns and structures, while simultaneously relating the different data-points. Supervised learning is usually more common in regression and classification cases where, in addition to the input, the algorithm also have access to the corresponding true output. Thus, the algorithm tries to find a correlation between the input and its true output [11]. The true output aid the algorithm to learn important details in the data. Supervised learning can thus be used for circuit design due to its capability of relating distinct S-parameters to matrix arrangements.

2.3.1 Convolutional Neural Network

Various algorithms are used to perform different types of prediction, where some are used for classification, for example classifying different animals, while others are used for regression, such as predicting stock prices. One common type of neural network (NN), which is used in this thesis, is a convolutional neural network (CNN) due to its reliable performance in image classification [12]. The general structure of a NN consists of input, hidden and output layers as displayed in Figure 2.3.

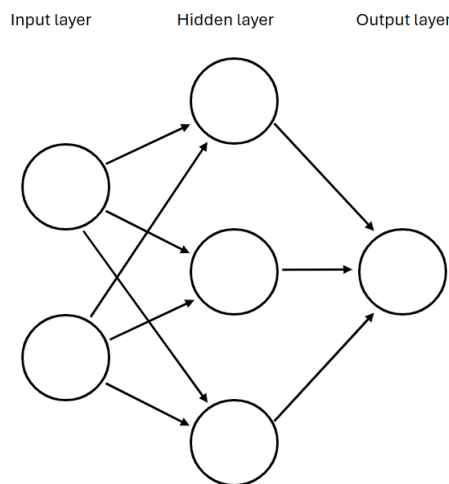


Figure 2.3: Display of layer structure of a CNN.

The number of hidden layers, and the number of interconnected nodes, illustrated as circles in Figure 2.3, is determined by the developer. Hidden layers are the backbone of the ML-model, as these layers weighs different features found in the input data based on its importance for the output. Thus, the number of hidden layers, the type of layers and the number of interconnected nodes are dependent on the task and

the desired precision. There are a couple of different types of layers used as hidden layers, such as convolutional layers and fully connected layers.

2.3.1.1 Convolutional layers

The core of a CNN are the convolutional layers, due to their capability to find and learn spatial features of an input. This is done with the aid of filters, more commonly known as kernels, that extracts the important features into a feature map [13]. The kernel can have different sizes, i.e., different width and height, in order to scan smaller parts of the image that is analyzed [14]. A feature map is created as the filters slide across the input, utilizing convolution [14]. Such a feature map may encompass multiple different important features such as edges and/or patterns.

2.3.1.2 Fully connected layers

Towards the end of the network there are usually some fully connected layers. The output layer usually consists of a dense layer with neurons matching the expected output dimension of the model. Depending on the task the network is supposed to solve, there might also be an activation function at the end. In the case of classification, this could be a softmax function that converts the raw output of the network into probabilities for each class. If it is a regression problem, the final layer might be a linear activation function if the predicted value from the last dense layer is supposed to be the output. If the value instead should fall within a certain range, the model can use a non-linear activation function. For example, if the model is supposed to predict values between -1 and 1 , the final layer could utilize a tanh activation function [13].

2.3.2 Model Evaluation

When creating a ML-model, the dataset is usually split into three parts, the training, validation and test set. The training set is used to train the model, and the validation set to evaluate how the training progress. Meanwhile, the test set is used after training to evaluate how well the final model performs. Depending on the task that is being solved, there are different types of metrics that can be used to assess the model performance. In supervised learning, the model can be evaluated by observing to what degree the model is able to predict the correct results. In the case of classification, such examination is done through accuracy, and precision metrics. This incorporates metrics such as F1-Score and confusion matrices. However, in regression problems, the output/prediction is an estimation of a certain value for the input data. Thus, in such cases a more suitable evaluation metric is an error measurement, such as Root Mean Square Error (RMSE) which is used in this thesis [15].

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (2.8)$$

The RMSE calculation, shown in Equation (2.8), fits this thesis well due to its capability to determine the error between the estimated and the true S-parameters.

Furthermore, to analyze the quality of the model, concepts such as overfitting and underfitting are helpful. Overfitting refers to the model memorizing the data in the training set, instead of identifying patterns in the input data. Thus, the model functions really well on the data it knows, but fails to perform as well on new, unseen data points. This can be seen by comparing training accuracy and validation accuracy. Cases of overfitting has high training accuracy and poor validation accuracy [16]. In the case of underfitting, the model is unable to find any sort of relationship between input and output. As a result, it will have poor accuracy for both the training and validation set [16].

2.4 Genetic Algorithms

Genetic algorithms (GA) are a type of evolutionary algorithms (EA) inspired by natural selection and the passing of genes [17], used to find optimized solutions to the desired problem. This is done in the same way as natural selection, in which the fittest individuals survive and pass their genetics to the next generation. Thus, GA are population based algorithms, meaning they consist of some arbitrary number of solutions, known as individuals, with different parameters, or genes. Each individual is then evaluated using a fitness function [18] which gives a higher score to better individuals. The algorithm then chooses some individuals to act as parents for the next iteration through different selection methods.

GA works by firstly initializing a completely random population that consists of several different individuals. These individuals have their own genes that represents that individuals' characteristics. For example, if the individuals are represented by matrices, the genes would be the elements in those matrices.

Following a successful iteration, each individual is evaluated by the fitness function. This function should be designed in such a way that it is able to separate a good solution from a poor one [17]. As a result, individuals that perform well are given a better score whilst poor performing individuals are given a worse score.

Individuals in the current generation are then selected to become parents for the next one, and this selection process can be done in several different ways. One example is the roulette wheel selection [19]. This method involves selecting an individual with a probability that is proportionate to its fitness value. Naturally, this means that a well performing individual has a greater chance of influencing the next generation than one that performs poorly.

When the parents have been chosen, the next step of the genetic algorithm is to create the individuals for the next generation. Once again there are several distinct methods, with one method being crossover. This mimics the way genetics works biologically by combining two parent solutions to produce new individuals [20]. Another popular approach which will be used in this thesis, is mutation [17], a method that consists of randomly introducing mutations in the selected parents and

thus ending up with new individuals. Following the reproduction, either some of the previous generation or the entire population is replaced with the newly created individuals. This process of selection, reproduction and replacement is repeated until a termination criteria is reached. This criteria can be either a desired score in an evaluation metric such as RMSE, or a limited number of iterations. A representative flow chart of this process is illustrated in Figure 2.4.

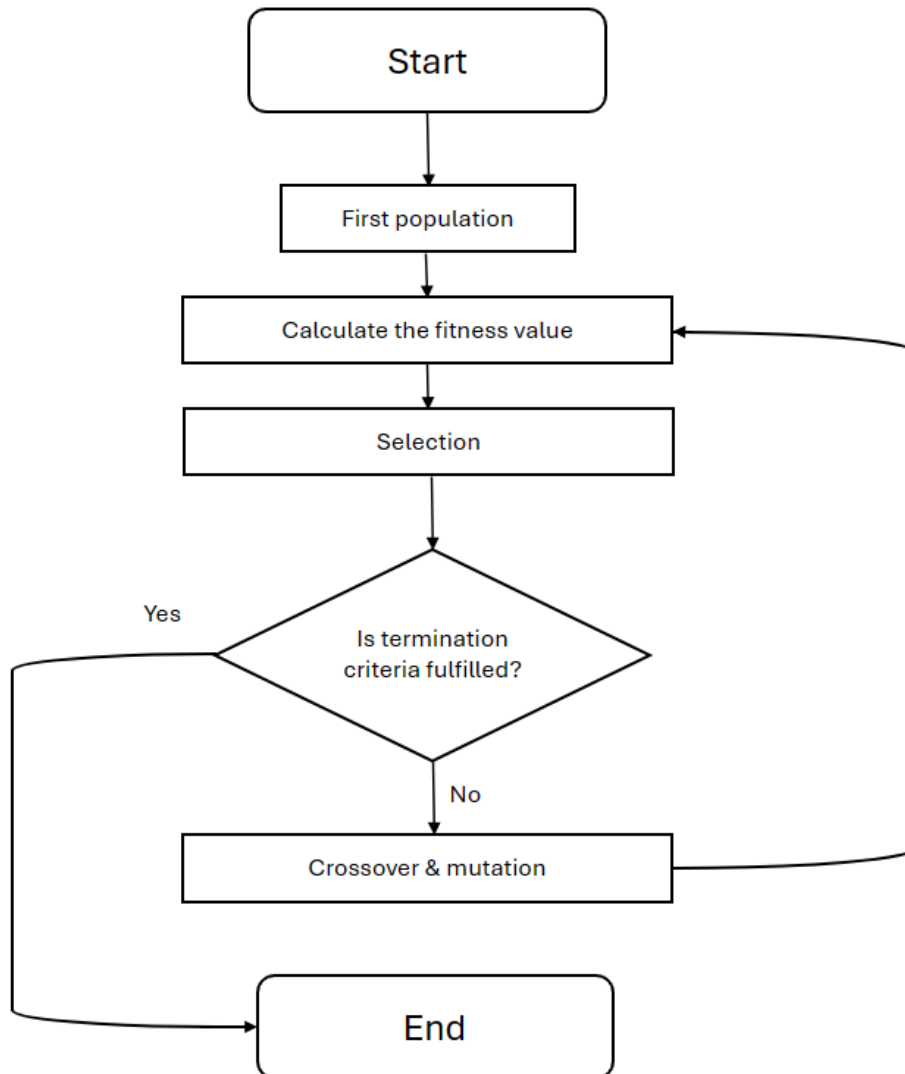


Figure 2.4: Flow chart for a genetic algorithm.

3

Development of Circuit Design Structure

This chapter will cover all design steps in the thesis, which includes the initial circuit built up in HFSS, S-parameter calculations and outcomes of different via fence simulations. These simulations are of the via fence's ability to prevent EMI in a circuit. The same circuit structure is then evaluated without the via fence to understand its impact.

3.1 Circuit structure in HFSS

The circuit structure used throughout the thesis is multilayered. It consists of a ground-plane of copper, a laminate substrate, and a copper array. The dimensions of the structure can be seen in Figure 3.1.

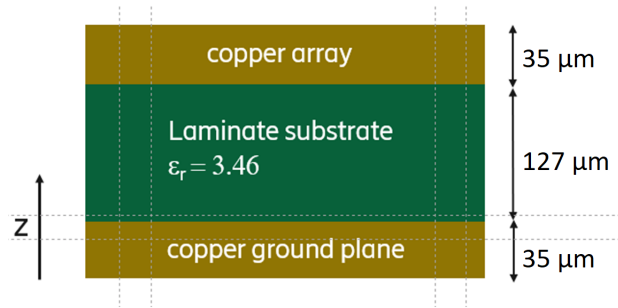


Figure 3.1: Circuit design's layer structure used in HFSS simulations.

The top layer, illustrated in Figure 3.1 is the 2-D pixel array, i.e., the $N \times N$ matrix where $N = 9$. Each metal cell is has the dimensions $500 \times 500 \mu m$ and does not use any padding. The circuit structure is illustrated in Figure 3.2 together with some indexation of ports and cells. On top of the laminate substrate in the circuit layer is also four connectors connecting the circuit with ports to allow the passage of electrical signals. The connectors are placed at the fifth cell in the circuit design, both along the y and x-axis. In order to obtain the datasets, multiple different circuits had to be simulated. To be able to produce multiple circuits, each 9×9 metal-matrix was randomly generated.

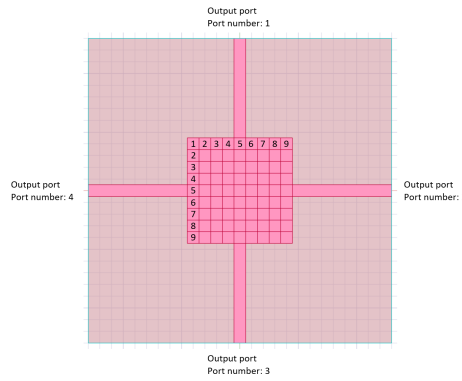


Figure 3.2: The indexation of the HFSS model.

This was done by setting the metal probability, p_{metal} , to $p_{metal} = 0.5$ for each cell in the matrix. Thus, each cell in the matrix has a 50 % chance of being metal and 50 % of not being metal. The presence of metal for each circuit over the whole dataset should thus follow a binomial distribution around 0.5. Such a distribution and probability factor, allowed for a large exploration of possible circuits. However, due to the random circuit design, a pathway between two ports was not guaranteed. This could lead to bad training data as multiple samples in the dataset would have S-parameters that were close to zero. To combat this issue a separate script received the matrix and randomly selected two I/O ports, and recursively created a metal pathway between the ports. As the structure had four possible I/O-ports, the script had a probability, p_{ports} set to $p_{ports} = 0.5$ to select an additional two ports to allow for circuits with multiple interconnected ports.

3.2 Via Fence Distancing

As mentioned in section 2.2, while creating a via fence the designer must closely determine different distances in the structure. Most importantly, the radiiuses of the via fence cylinders and the distancing between the fence and the circuit. The distance between the circuit and the via fence was based on manufacturing standards and multiple simulations for each distance. The via fence surrounded the entire circuit area, with equal s at each side of the circuit. All tests of the spacing in the circuit was based on initial requirements set by PCB guidelines, where the guidelines that were followed in this report are shown in Figure 3.3.

Mechanical Drilling	A	Minimum Outer Layer Annular Ring *	mm	0.125	0.075
	A	Minimum Inner Layer Annular Ring *	mm	0.125	0.1
	D	Minimum Mechanical Drilled Via Size	mm	0.2	0.15
	E	Minimum Spacing between Mechanically Drilled Holes	mm	0.4	0.35
	F	Estimated Finish Hole Size **	mm	0.15	0.1
	G	Minimum Inner Layer Copper Clearance	mm	0.225	0.2
	N	Maximum Non-Plated Mechanical Drill to Tent for Primary Drill Process	mm	5	6
	N	Maximum Non-Plated Mechanical Drill	mm	6.45	No Limit
	O	Minimum Non-Plated Mechanical Drill to Edge of Board	mm	0.25	0.2
	P	Minimum Copper to Non-Plated Mechanical Drill	mm	0.225	0.2
	* Above Minimum Annular ring numbers ensure tangency with no breakout. (If the minimum copper annular ring is required. Add 2X the annular ring requirement to the pad size. Pad size = $D + 2(A) + 2(\text{annular ring requirement})$. Example: $0.350 = 0.250 + 2(0.125) + 2(0.025)$ will ensure 0.025mm of the annular ring for a 0.250mm drill.) ** Estimated finished hole size is based on nominal plating thickness for standard via drill size. Calculation of DHS = $FHS + 0.075\text{mm}$. Adjustments to drill size are made based on hole wall copper plating thickness and surface finish type.				

Figure 3.3: The PCB guidelines used in the thesis.

With the requirement shown in Figure 3.3 in mind, different simulations were carried out to find the optimal structure of the circuit with a via fence. Theses simulations were performed on the test circuit seen in Figure 3.4.

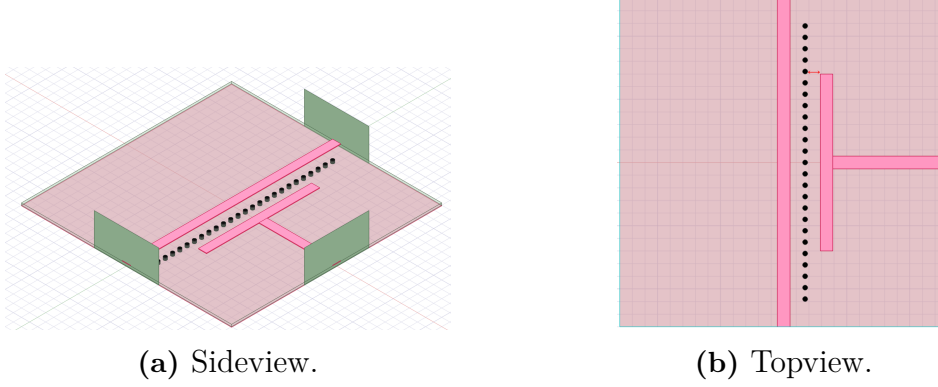


Figure 3.4: Illustration of the circuit used in the via fence evaluation.

The circuit consisted of three ports, two were connected via a transmission line and the third was separated from the other two by a via fence. During the simulations, the main focus was to minimize the leakage between the two connected ports and the third isolated port while still maintaining a compact circuit structure.

The two parameters that were examined were the distance between the edge of the via fence cylinders and the edge of the transmission line, as well as the diameter of the cylinders in the fence. These distances were examined in the same frequency range as the ML-model was to be trained for, i.e., the simulations were done in the frequency span of 1-10 GHz. The same simulations were also performed without the via fence, in order to see to what extent the via fence was able to decrease the effect of EMI. Each test scenario is analyzed based on how much of the unwanted signal that leaks to the separated connector despite the via fence. All cases can be seen in Table 3.1.

Table 3.1: Different spacings being tested for the via fence.

Test Number	Radius of Cylinder (mm)	Distance to Circuit (mm)
Test 1	0.150	0.25
Test 2	0.150	0.30
Test 3	0.150	0.35
Test 4	0.175	0.25
Test 5	0.175	0.30
Test 6	0.175	0.35
Test 7	0.200	0.25
Test 8	0.200	0.30
Test 9	0.200	0.35

The selected scenarios were based on a desire to keep the via fence as close to the circuit as possible to keep the structure compact. At the same time it should not surpass the cell padding distance mentioned in Equation (2.7). If the distance to the circuit increased too much, that would also mean that fewer modules would be able to fit together in a small space and thus the application capabilities would be reduced. The results are shown in Figures A.1, A.2, A.3 in *Appendix A*. It was desired to have couplings below -40dB while still having small distance for s .

In Subfigure A.1a, the coupling at 10 GHz is roughly -38 dB between the two isolated connectors. Meanwhile, in the case of Subfigure A.1c the final data point at the same frequency ends up at roughly -42 dB. When comparing these results to the results presented in Figure A.2, it is notable that a slight increase in cylinder radius aid prevention of EMI compared with a cylinder diameter of 0.15. As seen in Subfigure A.2a, the coupling at 10 GHz is about -40 dB. Meanwhile, using a distance of 0.35 mm as shown in Subfigure A.2c results in a coupling of roughly -43 dB. There is a diminishing return when increasing the distance compared to the previous test. The test results for the final radius size, shown in Figure A.3, show that the data point at 10 GHz has decreased slightly and is now almost exactly -40 dB in Subfigure A.3a. The same phenomena can be seen in both Subfigure A.3b and Subfigure A.3c, as the values have only been lowered slightly. The final results from these tests hints at the fact that further increasing the cylinder size will give diminishing returns for preventing EMI. As such the cylinder size was chosen to be $R = 0.20$ mm, and the distance between the circuit and via fence was decided to be $s = 0.25$ mm. The reason for choosing the smaller distance even though the larger distances resulted in a somewhat lower interference was due to the fact that the difference in preventing EMI was negligible, but also that the circuit should be as compact as possible. In addition to these trials, a comparison was done between a circuit with a via fence and one without it. This comparison was done with the chosen distance from the previous trial, i.e., $R = 0.20$ mm and $s = 0.25$ mm. In these tests, the same test circuit was used, Figure 3.4. These tests were conducted to investigate to what extent the via fence could prevent EMI. The results are shown in Figure A.4.

As illustrated by Figure A.4, there is a roughly 2 dB difference in isolation between the two circuits. This is not a huge difference, but it might still have an impact on the modular circuit as they will be surrounded by the via fence. By incorporating via fences in the circuit design, the module-based design concept could potentially achieve better results due to its tolerance against interference.

3.3 Assembly of Modules

In order to assemble multiple circuits into one modular circuit, there are some fundamental elements to keep in mind. These elements are vital, and include that each circuit should have the same reference frequency ranges and Z_0 . The first step is to decide the number of modules that will be used, which is highly important as each module will have an effect on the final S-parameters. Throughout this thesis, the number of modules $M = 4$, will be structured in a 2×2 configuration. This is

illustrated in Figure 3.5.

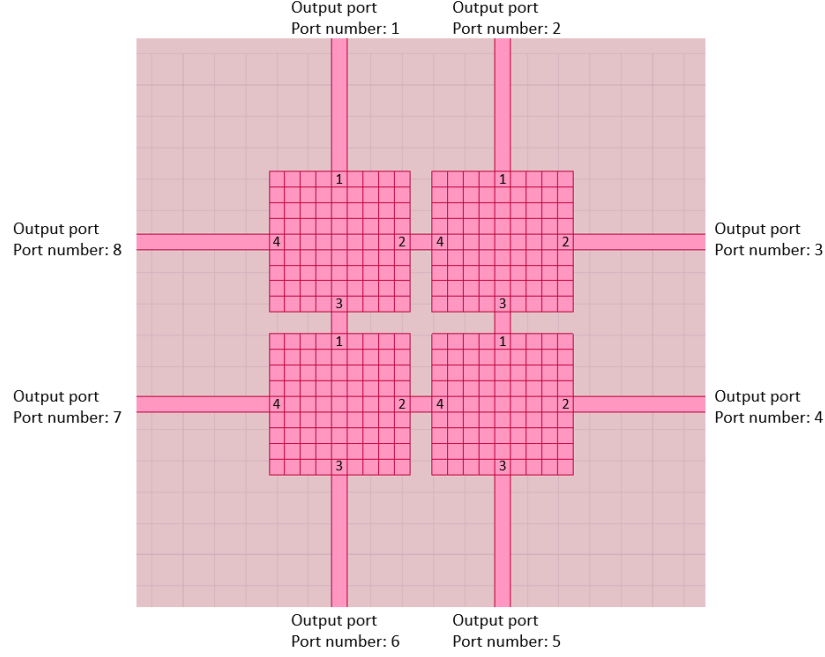


Figure 3.5: Notation of the ports for each module and notation of the ports for the complete modular circuit.

As seen in Figure 3.5, each module has its own circuit port numbers, i.e., four designated I/O port numbers. When combining a pair of modules, one port from each circuit are connected together. This type of connection will no longer be considered an I/O port, instead being seen as a part of the circuit. Thus, the final number of I/O ports decreases, in this case to $n = 8$ and $n_{connections} = 4$. During the calculation, each module has its individual, 4×4 S-matrix which is needed for the final calculation of the modular circuit's S-parameters. The circuits' S-parameters will be indexed with the module number m from here on, to refer to each circuit's S-parameters as shown in Equation (3.1).

$$\mathbb{S}_m = \begin{bmatrix} S_{m,11} & S_{m,12} & S_{m,13} & S_{m,14} \\ S_{m,21} & S_{m,22} & S_{m,23} & S_{m,24} \\ S_{m,31} & S_{m,32} & S_{m,33} & S_{m,34} \\ S_{m,41} & S_{m,42} & S_{m,43} & S_{m,44} \end{bmatrix} \quad (3.1)$$

Before combining each module's respective S-matrix, seen in Equation (3.1), the circuit's S-matrices are converted to admittance matrices to facilitate an easier analysis, utilizing Equation (2.2). The analysis becomes easier as the admittance for multiple connected modules can be added together. $\mathbb{Y}$ has the same size as the S-matrix that was converted, i.e., 4×4 as shown in Equation (3.2).

$$\mathbb{Y}_m = \begin{bmatrix} Y_{m,11} & Y_{m,12} & Y_{m,13} & Y_{m,14} \\ Y_{m,21} & Y_{m,22} & Y_{m,23} & Y_{m,24} \\ Y_{m,31} & Y_{m,32} & Y_{m,33} & Y_{m,34} \\ Y_{m,41} & Y_{m,42} & Y_{m,43} & Y_{m,44} \end{bmatrix} \quad (3.2)$$

The conversion admittance matrices allow for the creation of a $\mathbb{Y}_{mod}$ consisting of the elements from the modules' $\mathbb{Y}$ matrices. The total number of connections and I/O ports in $\mathbb{Y}_{mod}$ will be lower than the total number of ports for the modules as some ports are shared. Thus, Equation (2.3) has to be used with $M = 4$, $n = 4$ and $n_{connections} = 4$ leading to the dimension of $\mathbb{Y}_{mod}$ being $D = 12$. Figure 3.6 showcases the new port configuration, where each intermediate port has been assigned a new number. These numbers are added to facilitate the combination of circuits mathematically.

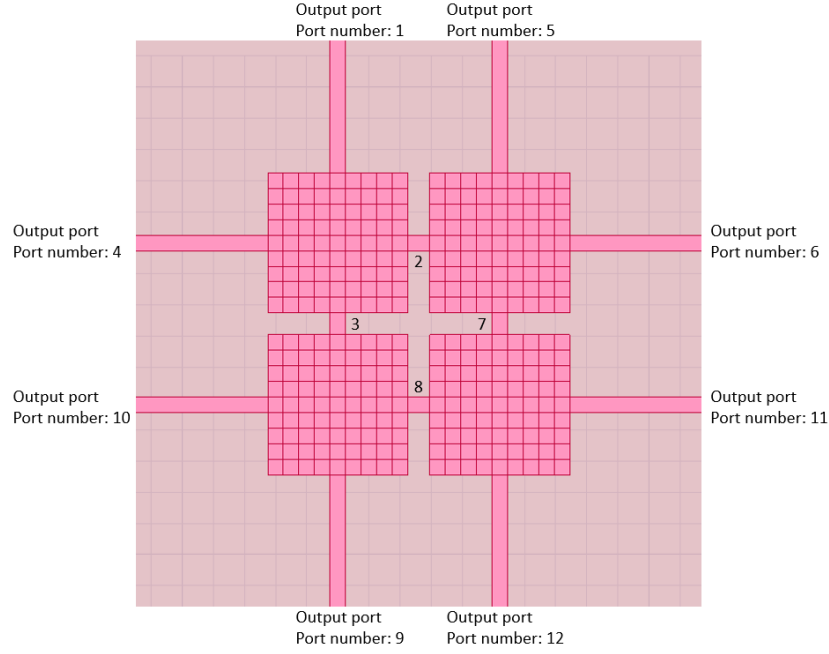


Figure 3.6: Intermediate port notation during S-parameter calculation.

The S-parameters from the modules' $\mathbb{Y}$ matrices are then inserted into $\mathbb{Y}_{mod}$. This matrix has a size of $D \times D$, as shown in Figure 3.6, some ports will share S-parameters from the modules. The resulting matrix can be seen in Equation (3.3).

(3.3)

$$\mathbb{Y}_{mod} = \begin{bmatrix} Y_{1,11} & Y_{1,12} & Y_{1,13} & Y_{1,14} & 0 & 0 & 0 & 0 & 0 & 0 \\ Y_{1,21} & Y_{1,22} + Y_{2,44} & Y_{1,23} & Y_{1,24} & Y_{2,41} & Y_{2,42} & Y_{2,43} & 0 & 0 & 0 \\ Y_{1,31} & Y_{1,32} & Y_{1,33} + Y_{3,11} & Y_{1,34} & 0 & 0 & 0 & Y_{3,12} & Y_{3,13} & Y_{3,14} \\ Y_{1,41} & Y_{1,42} & Y_{1,43} & Y_{1,44} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{2,14} & 0 & 0 & Y_{2,11} & Y_{2,12} & Y_{2,13} & 0 & 0 & 0 \\ 0 & Y_{2,24} & 0 & 0 & Y_{2,21} & Y_{2,22} & Y_{2,23} & 0 & 0 & 0 \\ 0 & Y_{2,34} & 0 & 0 & Y_{2,31} & Y_{2,32} & Y_{2,33} + Y_{4,11} & 0 & 0 & 0 \\ 0 & 0 & Y_{3,21} & 0 & 0 & 0 & Y_{4,41} & Y_{4,14} & 0 & 0 \\ 0 & 0 & Y_{3,31} & 0 & 0 & 0 & 0 & Y_{3,22} + Y_{4,44} & Y_{3,23} & Y_{3,24} \\ 0 & 0 & Y_{3,41} & 0 & 0 & 0 & 0 & Y_{3,32} & Y_{3,33} & Y_{3,34} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & Y_{3,42} & Y_{3,43} & Y_{3,44} \\ 0 & 0 & 0 & 0 & 0 & 0 & Y_{4,21} & Y_{4,24} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Y_{4,31} & Y_{4,34} & 0 & 0 \\ & & & & & & & & & Y_{4,12} & Y_{4,13} \\ & & & & & & & & & Y_{4,42} & Y_{4,43} \\ & & & & & & & & & 0 & 0 \\ & & & & & & & & & Y_{4,22} & Y_{4,23} \\ & & & & & & & & & Y_{4,32} & Y_{4,33} \end{bmatrix}$$

$\mathbb{Y}_{mod}$ in Equation (3.3) could be converted back to the S-matrix if desired, however this does not account for unconnected I/O-ports or short circuited ports. Throughout this thesis each unused port was handled as unconnected, and not short circuited. In order to truncate the matrix, i.e., eliminate the unconnected ports, a conversion to Z-parameters is performed. Thus, the Equation (2.4) can be used to convert the $\mathbb{Y}_{mod}$ matrix to a $\mathbb{Z}_{int}$ matrix. An I/O port pair then has to be chosen whereas all the rows and columns are unconnected. The unconnected ports and the intermediate ports from Figure 3.6 are deleted from the $\mathbb{Z}_{int}$ to get $\mathbb{Z}$. An example of this is showed in Figure 3.7, where port 1 and 5 has been chosen.

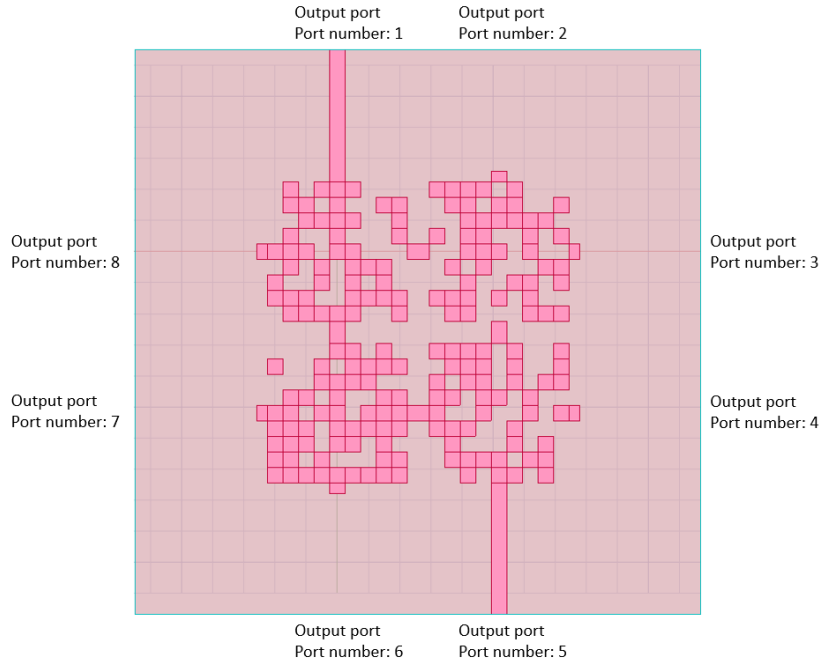


Figure 3.7: Modular circuit with unconnected ports.

The representative $\mathbb{Z}$ matrix of the circuit in Figure 3.7 is shown in Equation (3.4).

$$\mathbb{Z} = \begin{bmatrix} Z_{11} & Z_{15} \\ Z_{51} & Z_{55} \end{bmatrix} \quad (3.4)$$

The matrix in Equation (3.4) can be transformed back into the wanted $\mathbb{S}$ by using Equation (2.5). $\mathbb{S}$ will have the same size as $\mathbb{S}_{base}$ due to the modular circuit only having two I/O ports. The resulting $\mathbb{S}$ matrix is shown in Equation (3.5).

$$\mathbb{S} = \begin{bmatrix} S_{11} & S_{15} \\ S_{51} & S_{55} \end{bmatrix} \quad (3.5)$$

4

Development of Machine Learning Model and Genetic Algorithm

This section presents the complete process of developing a ML-model for the issue at hand. Featuring in this section will be the development of the dataset, i.e., important preprocessing details and augmentations. Furthermore, it will go through the process of finding the two optimized ML-models, as well as the procedure the genetic algorithm.

4.1 Data Generation and Preprocessing

Data generation in this research is mainly done through numerous EM simulations in order to create the datasets. Two datasets were generated, one for a circuit model without a via fence and another for a circuit model with a via fence. The layout of the metal pattern and the corresponding S-parameters were used to train the ML-model. Each circuit was represented by an array which was the input to the ML-model. An illustration of an equivalent matrix is shown in Figure 4.1.

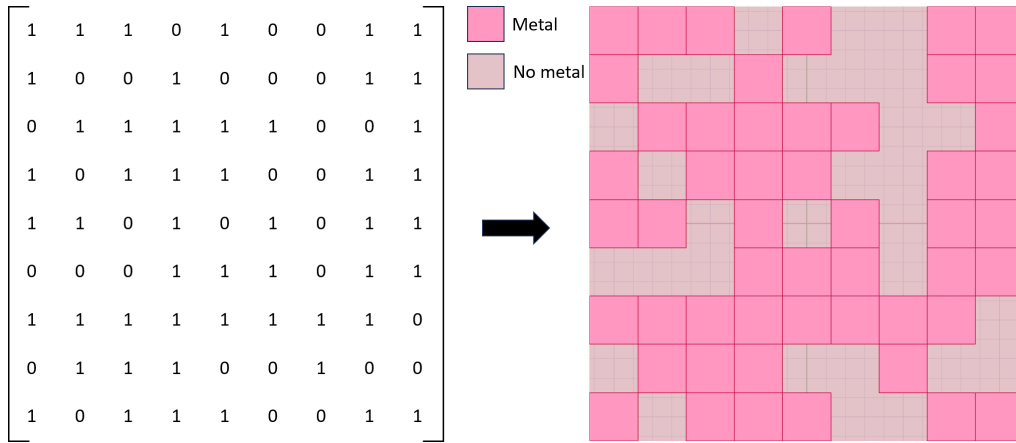


Figure 4.1: Matrix representation of the circuit.

The datasets have the same size, consisting of 30 000 randomly generated matrices, all randomly generated. The difference between the datasets lie solely in their S-parameter data due to the impact of the via fence. Ahead of any simulation however, each circuit was reviewed with a script to evaluate if it had a pathway between at least one I/O port pair. If there were no connections, the script had a probability of

$p_{\text{pair}} = 0.95$ of connecting two randomly chosen ports. Some circuits were allowed to remain unconnected so that the model could understand the implication of lacking a connection between I/O-ports. The circuits were then simulated to find their S-parameters. After the simulations, the different circuits were augmented to increase the size of the datasets and allow for more training data. This also ensures that the model is not biased towards connections between certain ports. The augmentation consisted of both rotations and flips of the circuit matrices. The corresponding S-parameters were also transformed to ensure that they end up at the correct port connections. A few examples of the data augmentations are presented in Figure 4.2.

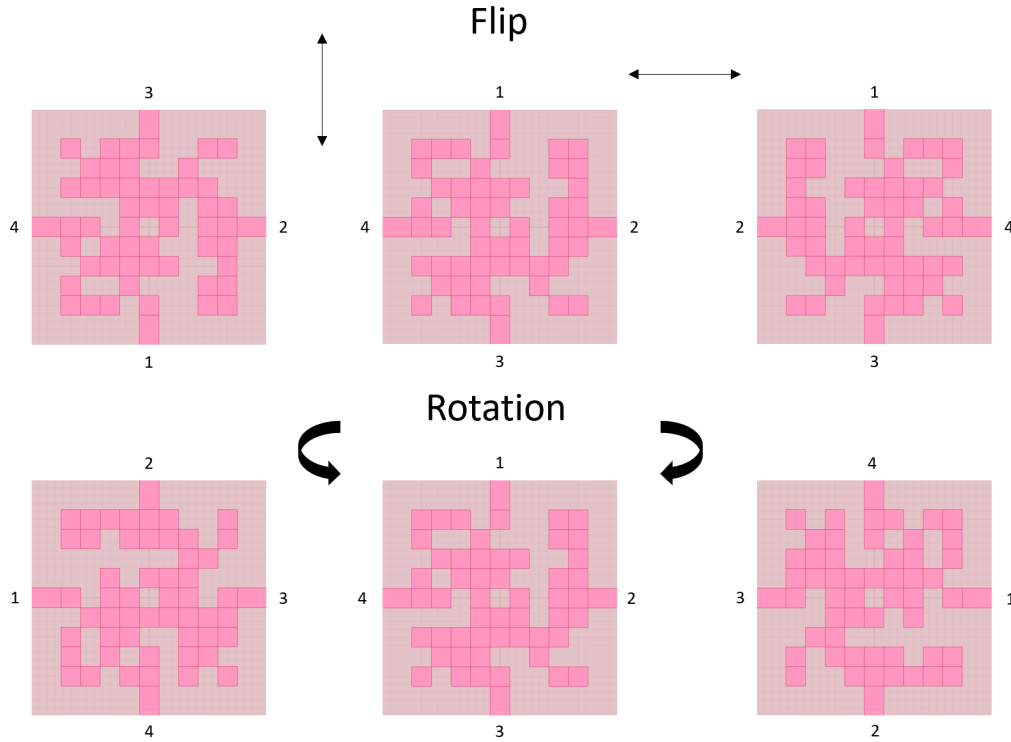


Figure 4.2: Augmentations that were conducted.

Rotating the matrices yields an additional three circuits, whilst flipping introduces an additional two. A number of combinations of rotations and flipping also introduced an additional two unique circuit matrices. Hence, for each original circuit in the dataset there will be seven augmented versions, leading to a total of 240 000 circuits and S-parameter combinations. The details of the augmentations can be seen in Table 4.1.

Table 4.1: Details for the augmentations.

Configuration	Number of Simulations	Constellations for each circuit	Final dataset size
Non-via fence	30 000	8	240 000
With via fence	30 000	8	240 000

4.2 Model Selection and Training Accuracy

When the dataset had been prepared, the next step was to look at what type and architecture the ML-model should have. A simple CNN architecture was developed and trained in order to evaluate the initial performance in predicting a circuit's S-parameters. The composition of the network could then be modified to find a well performing model for accurate S-parameter prediction. During training, the model parameters are adjusted to minimize a loss function. This is performed by an optimization algorithm using gradient descent, in this thesis the Adaptive Moment Estimation (ADAM) algorithm. During this phase the focus was to tune the hyperparameters of the model to improve its performance.

4.2.1 Architecture selection

For all different architectures, the dimensions of the input as well as output layer remained the same. The input consisted of a matrix with 9×9 elements. As for the output layer, it was of the same size as the number of S-parameters for a four-port circuit, i.e. the elements shown in Equation (3.1). However, as each individual S-parameter is a complex number, a real and an imaginary number, the output for each frequency consisted of 32 neurons and not 16. The 10 frequency points resulted in the output layer consisting of 320 neurons.

The initial model consisted of six convolutional layers with 16 filters with the following kernel sizes: [4, 4, 3, 3, 2, 2]. Following this it had three dense layers, each with 16 neurons each, and then the output layer. To avoid overfitting, the model utilized batch normalization and dropout of 15% in both the convolutional and the fully connected layers. Both layers utilized the Exponential Linear Unit (ELU) as activation function, whilst the final output layer used a linear activation function. The performance on the testset was evaluated using RMSE, and this model was then used as a baseline to find a better model architecture by looking for a lower RMSE. The RMSE values for the baseline model can be seen in Table 4.2

Table 4.2: The RMSE values for the baseline model.

Mean RMSE Value	Lowest RMSE Value	Highest RMSE Value
0.11555	0.02902	0.30647

Initial tests were performed in an iterative manner, where parts of the architecture were changed and the impact evaluated based on the RMSE score. The fine-tuning mainly consisted of trial-and-error, by changing individual parameters such as the number of layers or neurons.

The first parameter that was investigated was the number of neurons in the dense layers, and the results can be seen below in Table 4.3. It can be seen that increasing the number of neurons of the model leads to a slight increase in performance. The

improvement is more prominent at first, and then there is a diminishing return as more neurons are added.

Table 4.3: RMSE scores for different numbers of dense layers neurons.

Neurons	Mean RMSE Value	Lowest RMSE Value	Highest RMSE Value
2	0.21479	0.09300	0.39666
4	0.16145	0.06757	0.41743
8	0.14217	0.31529	0.05539
16	0.11319	0.02757	0.27672
32	0.08573	0.02212	0.27811
64	0.06296	0.01525	0.24808
128	0.04674	0.00890	0.23671
256	0.03467	0.00608	0.20269
512	0.02595	0.00383	0.16062
1024	0.02106	0.00383	0.20109

The results in Table 4.4, show that increasing the number of dense layers does not result in a better model. The best model in this case is the one utilizing a single dense layer indicating that having only one fully connected layer at the end is the optimal choice.

Table 4.4: RMSE scores for different numbers of dense layers.

Dense Layers	Mean RMSE Value	Lowest RMSE Value	Highest RMSE Value
1	0.09529	0.02489	0.26387
2	0.11152	0.02722	0.30725
4	0.12207	0.03554	0.31822
8	0.12930	0.03532	0.35759
16	0.16149	0.05785	0.45078
32	0.23496	0.11793	0.50473
64	0.26342	0.15049	0.48679

Table 4.5 shows the results when using different numbers of convolutional layers. Similarly to when increasing the number of neurons in the dense layers, these trials show a steady improvement in performance that eventually stagnates as the number of layers grows.

Table 4.5: RMSE scores for different number of convolutional filters.

Convolutional Filters	Mean RMSE Value	Lowest RMSE Value	Highest RMSE Value
1	0.17035	0.04428	0.47627
2	0.15009	0.04227	0.44483
4	0.14358	0.04009	0.41563
8	0.12729	0.03743	0.33077
16	0.11880	0.03052	0.28562
32	0.10958	0.02897	0.26679
64	0.10572	0.02501	0.25009
128	0.10027	0.02793	0.27908

In Table 4.6 the impact of different number of convolutional layers with varying kernel sizes are shown. In this case, there are three separate cases that result in a similar RMSE score with a slight difference in lowest and highest RMSE. These results indicate that after reaching a certain number of convolutional layers with certain kernel sizes, the model is able to capture most of the features of the data. As such, further increase of the number of layers and kernel sizes will not result in additional improvements, but only making the model more complex.

Table 4.6: RMSE scores for different number of convolutional layers and kernel sizes.

Kernel size of layer	Mean RMSE Value	Lowest RMSE Value	Highest RMSE Value
5, 5	0.12009	0.03364	0.31816
5, 5, 4, 4	0.11651	0.02621	0.29923
5, 5, 4, 4, 3, 3	0.11537	0.02896	0.28447
5, 5, 4, 4, 3, 3, 2, 2	0.10871	0.02957	0.26612
6, 6, 5, 5, 4, 4, 3, 3, 2, 2	0.10961	0.02811	0.28105
7, 7, 6, 6, 5, 5, 4, 4, 3, 3, 2, 2, 1, 1	0.10833	0.02503	0.28067

4.2.2 Evaluation of the Model

During the evaluation of the ML-models, both the RMSE as well as the training history were considered. The parameters that were used in the baseline model are listed in Table 4.7.

Table 4.7: The structure of the baseline model.

Kernel size of layers	Convolutional filters	Dense layers	Neurons in dense layers
4, 4, 3, 3, 2, 2	16	3	16

Furthermore, the training graphs for the baseline model can be seen below in Figure 4.3, with the left figure showing the results during 100 epochs, whilst the right hand one is zoomed in on the first 10. As illustrated by both figures, the loss for training and validation quickly reduces in the first couple of epochs and then stabilizes. They also show no trace of overfitting as the validation curve follows the training curve. Thus the baseline model seems to be able to capture the underlying relation between input and output, although the poor RMSE seen in Table 4.2 implies that it can be improved.

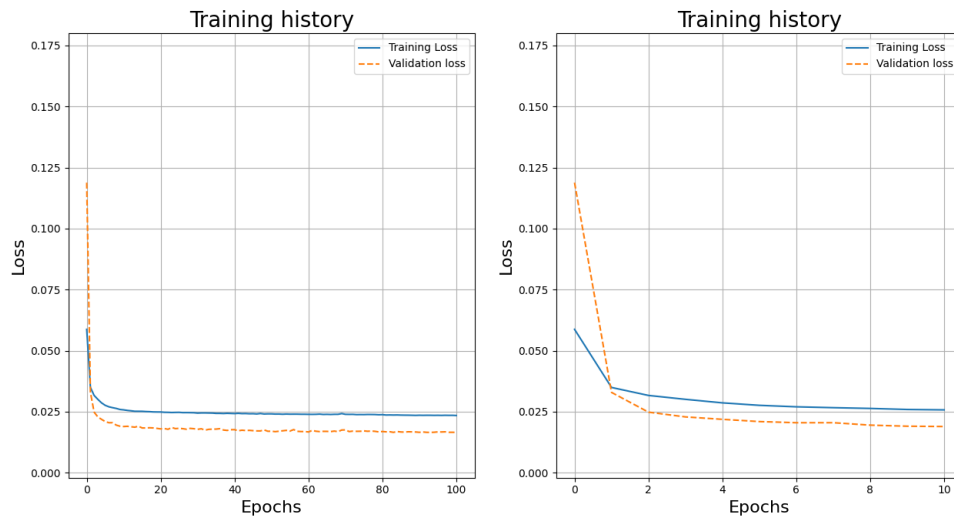


Figure 4.3: The training history of the baseline model.

Using the results from the previous section, the optimized non-via fence model ended up having the parameters shown in Table 4.8.

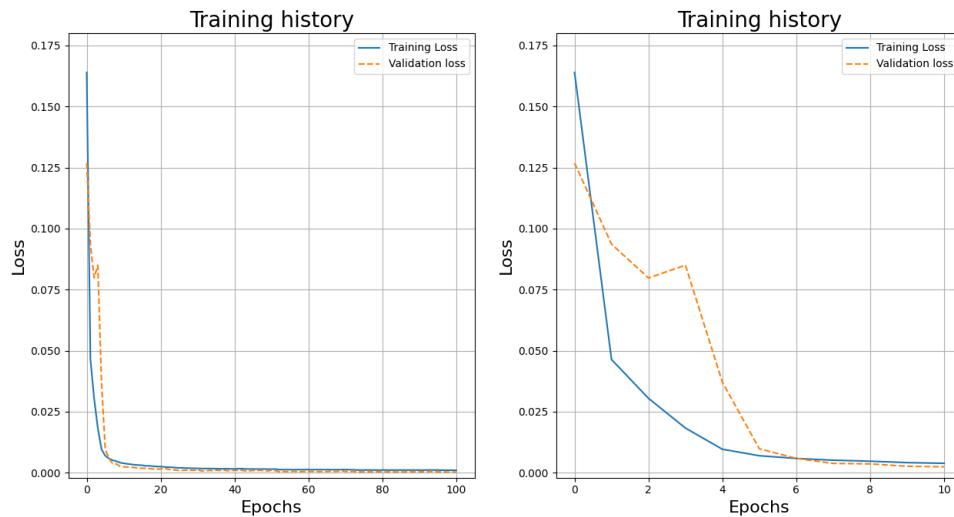
Table 4.8: The structure of the optimized non-via fence model.

Kernel size of layers	Convolutional filters	Dense layers	Neurons in dense layers
5, 5, 4, 4, 3, 3, 2, 2	128	1	768

The loss curves for the model are shown in Figure 4.4. Just like the baseline model, these curves show no trace of overfitting, as the validation curve closely follows the training curve. However, this model has a better performance than the baseline model as it has lower loss for both training and validation. This can also be seen in Table 4.9 as the mean RMSE value is roughly 10 times lower compared to the baseline model.

Table 4.9: The RMSE values for the best model for the non-via fence dataset.

Mean RMSE Value	Lowest RMSE Value	Highest RMSE Value
0.01075	0.00217	0.14027

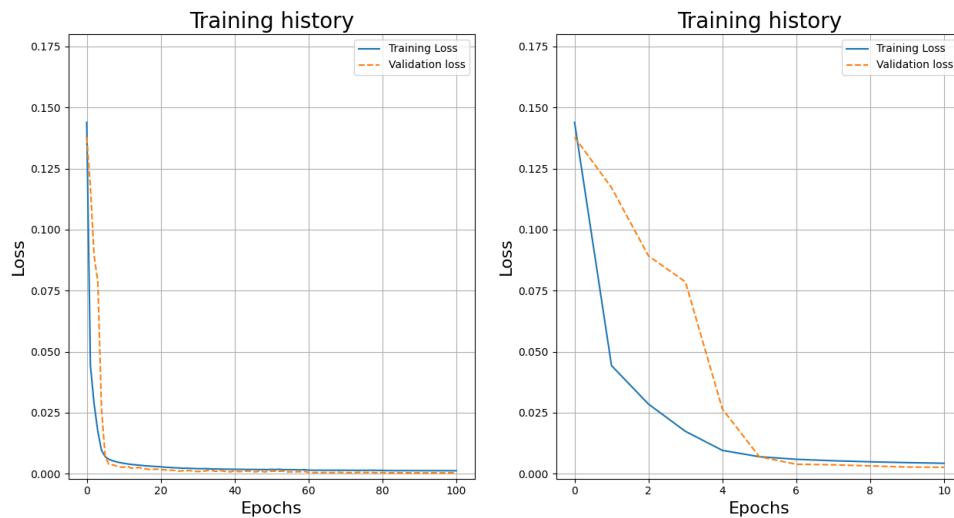
**Figure 4.4:** The training history of the best model without via fence.

The architecture for the optimized via fence model was almost identical to the non-via fence model, and its parameters can be seen in Table 4.10.

Table 4.10: The structure of the optimized via fence model.

Kernel size of layers	Convolutional filters	Dense layers	Neurons in dense layers
5, 5, 4, 4, 3, 3, 2, 2	128	1	512

The only difference was that the dense layer consisted of 516 neurons instead, with the reason being that the validation loss showcased a number of outliers, even through multiple training iterations. Lowering the neurons prevented this phenomena from appearing. The training history for the model is presented in Figure 4.5. Just like the other model it shows no trace of over- or underfitting as the validation loss is following the training loss. As for the RMSE, the mean is somewhat higher when compared with the non-via fence model, but this could be explained by the lower complexity of the model. Since the difference is small, and the loss curves do not have any outliers, thus hinting that the model is more stable, it was decided to keep the lower number of neurons.

**Figure 4.5:** The training history of the best model with via fence.

The final results for the via fence dataset can be seen in Table 4.11.

Table 4.11: The RMSE values for the best model for the via fence dataset.

Mean RMSE Value	Lowest RMSE Value	Highest RMSE Value
0.01105	0.00240	0.08864

4.3 Genetic Algorithm

The GA implementation is constructed in the manner showed in Figure 2.4. Initially, the algorithm is started by randomly generating candidate circuits that will be the first parents circuits in the process, i.e., the initial population. These circuits are four different 9×9 matrices combined into a 2×2 modular circuit. Due to the structure of the output layer of the ML-model, it can only produce a prediction on each 9×9 matrix individually, so the S-parameters of the full circuit have to be obtained as described in section 3.3. The fitness function used in the GA is the same as in the ML-model training, i.e., RMSE to evaluate the candidate circuits. RMSE is calculated by comparing the S-parameters for the candidate circuits with the targeted S-parameters. The circuits with the lowest RMSE are selected for the next iteration. However, before proceeding these circuits are evaluated to see if they have either reached a desired RMSE or if a maximum number of iterations has been reached. The initial population, the number of circuits passed on to the next iteration and the number of iterations are shown in Table 4.12.

Table 4.12: Constants in the genetic algorithm.

Initial Population	Best Circuits	Iterations	Termination RMSE Score
780	30	100	0.001

If neither termination requirement is met, the algorithm continues to the mutation process. The mutation process begins by duplicating the chosen circuits until reaching the initial population size. The circuits are then randomly mutated according to Equation (4.1), where $\mathcal{R} = \mathcal{U}[0, 1)$ and $\mathcal{N}$ is the current iteration. Early in the iteration process, the mutation rate is higher to allow for a larger exploration of possible circuit configurations. The rate of mutation will then slowly decrease over time due to its dependency on the current iteration.

$$0.01 + 0.25 \cdot \frac{\mathcal{R}}{\log_{10}(\max(10, \mathcal{N}))} \quad (4.1)$$

The mutation is a bit-wise mutation where each element in the circuit array has the probability, found in Equation (4.1), of being flipped. Thus, an element in the circuit matrix could be altered from a 1 to a 0, which will change the prediction of the S-parameters of the candidate circuit. The entire process of the GA is then repeated until either termination criteria is met, as shown in Figure 4.6.

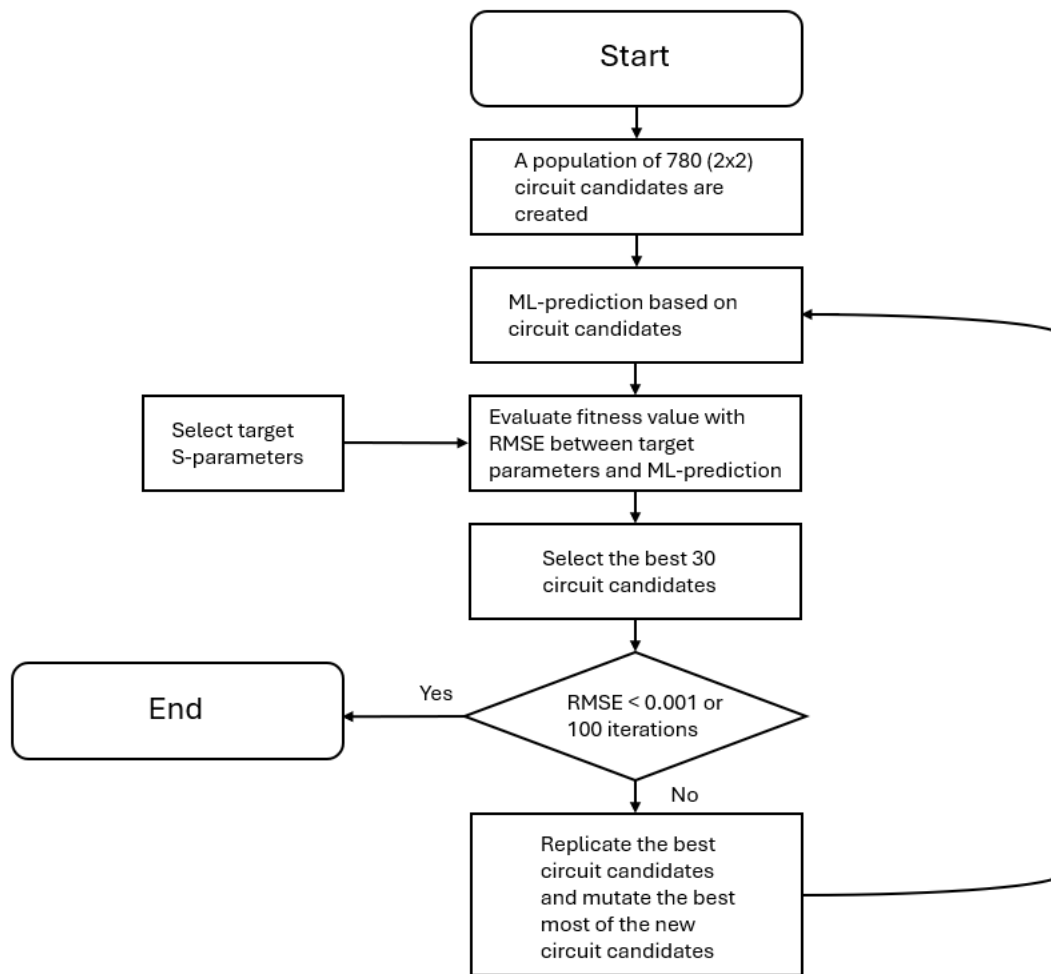


Figure 4.6: Illustration of the genetic optimization process for a circuit consisting of 2×2 circuit modules.

5

Results

This chapter presents the results found for each research question mentioned in section 1.2. This includes an assessment of the S-parameters for the genetically optimized circuits compared to the target S-parameters. Furthermore, the genetically generated circuits were simulated in HFSS to verify the results and compare them with the GA's predicted S-parameters. Together with the representation of the S-parameters, the impact of the interference between the modules will be assessed by comparing the results obtained using the circuit models with and without via fence. Lastly, the time efficiency of the concept was assessed through an examination on the simulation time, training time and optimization time.

5.1 Scattering Parameter Depiction

In order to assess the modular concept's representation of the desired S-parameters, different metrics and tests were constructed. Multiple test circuits were created, where each had its own target S-parameters. The target parameters for one modular circuit was created using four circuits from the test set which had a connection between the two selected I/O ports in the modular circuit. The process was repeated 100 times with different target parameters. To enable a direct comparison, the same S-parameter targets were used for the non-via fence case as well as the via fence case. The circuits generated with the genetic algorithm were then simulated in HFSS to compare the genetic algorithm's guess of the S-parameters with the simulated S-parameters. The average RMSE for the simulated and optimized S-parameters were then calculated for the 100 circuits. Results are further showcased by a randomly chosen example circuit from the tests, illustrating the absolute values for port 1 and port 5, i.e. $|S_{11}|$, $|S_{15}|$, $|S_{51}|$ and $|S_{55}|$, for the target, predicted and simulated S-parameters. Of most interest is the representation of the transmission S-parameter, $|S_{15}|$ and thus it will mainly be used in the results. However, the S-parameters for the reflection for both the input and output port are shown in the *A Appendix*.

5.1.1 Full Frequency Range Analysis

Figure 5.1 showcases the absolute values of the target parameters, as well as the predicted S-parameters from both the via fence test and the test without a via fence. For the same test example, the absolute value for the reflection port of the simulated circuits can be seen in Figure A.6. The respective transmission S-parameters, $|S_{51}|$, are as expected nearly identical to $|S_{15}|$ and thus are also in *A Appendix* and can be

seen in Figures A.9, A.10.

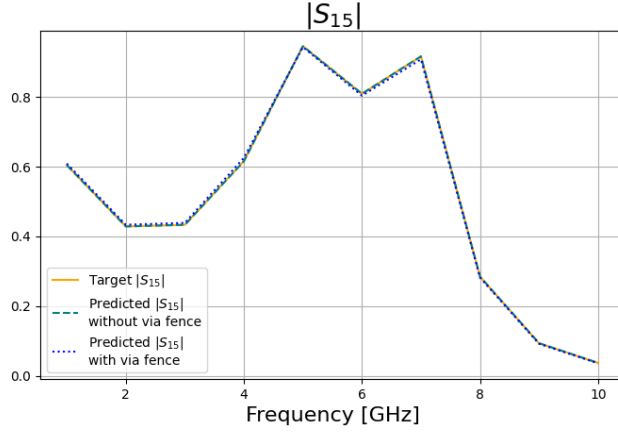


Figure 5.1: Plot of $|S_{15}|$ for prediction and target for both models.

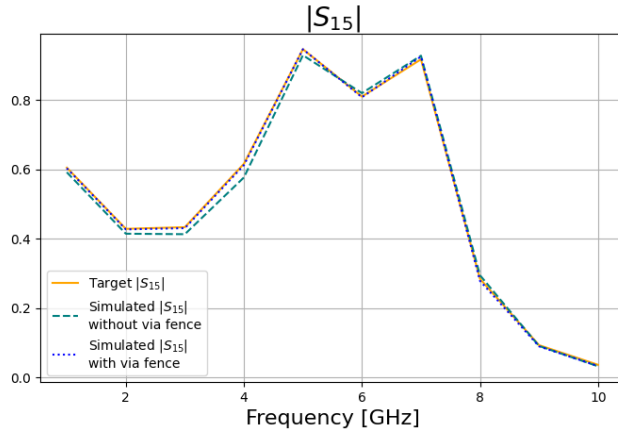


Figure 5.2: Plot of $|S_{15}|$ for simulation and target for both models.

Notably, seen in Figure 5.1 is that the S-parameters of the generated circuits follows the target S-parameter closely for every frequency point in the 1-10 GHz span, and both cases seem equally good. However, the results shown in Figure 5.2 indicates that the circuit generated using the circuit model without via fence is not as good as the circuit created using a circuit model with a via fence. The non-via fence case manages to capture the pattern of the target, however the actual values show a noticeable deviation. In this test scenario, the predicted RMSE for the non-via fence model is lower than the simulation RMSE for both transmission and reflection. However, the opposite is true in the via-fence case where the RMSE scores are higher for the prediction as shown in Table 5.1. A positive thing to note in Table 5.1, is that the predicted and simulated RMSE for $|S_{15}|$ and $|S_{51}|$ are equal, meaning that both ML-models have understood the underlying pattern of $S_{15} = S_{51}$.

Table 5.1: RMSE for prediction and simulation from both models for different $|S_{ij}|$ parameters.

Parameter	Model	Prediction RMSE	Simulation RMSE
$ S_{11} $	Non-via fence	0.00599	0.02478
	With via fence	0.00861	0.00428
$ S_{15} $	Non-via fence	0.00538	0.01838
	With via fence	0.00836	0.00728
$ S_{51} $	Non-via fence	0.00538	0.01838
	With via fence	0.00836	0.00728
$ S_{55} $	Non-via fence	0.00903	0.01658
	With via fence	0.01251	0.00317

However, this scenario does not represent the entire 100 circuit test batch. To summarize the performance, the average RMSE for all 100 circuits have been calculated and can be seen in Table 5.2.

Table 5.2: Mean RMSE value for 100 circuits.

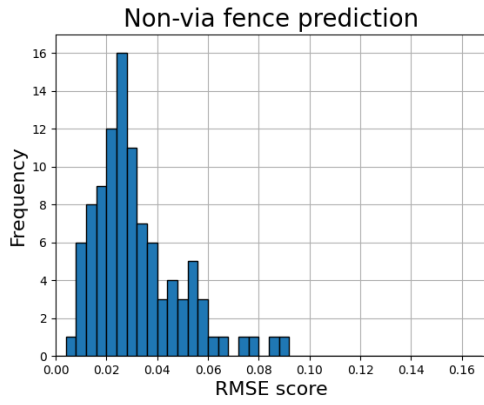
Configuration	Mean prediction RMSE Value for 100 circuits	Mean simulation RMSE Value for 100 circuits
Non-via fence	0.03184	0.05934
With via fence	0.03310	0.04677

The simulated RMSE values are slightly higher than the predicted RMSE values which is not surprising since neither ML-models is able to perfectly predict the S-parameters. During the genetic optimization, the model considers its guess to be a perfect representation of the S-parameters, and tries to perfectly fit it with the target. The error in the model’s accuracy thus manifests itself when the generated circuits are simulated. However, what can be noted is that the difference in mean RMSE for the via fence case is notably closer to its prediction than it is for the non-via fence case. Despite the two models having almost the same average RMSE for the ML-prediction, as illustrated in Table 5.2, the via fence model has a notably lower mean RMSE from the simulations. This difference might be due to fortuitous mutations in the GA, but is most likely a consequence of the via fence’s ability to isolate the modules from each other. The variance of the RMSE scores also shows indications of robustness of the performance of the circuits. The variance and standard deviation are shown in Table 5.3 and show that the models are rather confident for many different circuit structures, as the variance for the predicted RMSE and the simulated RMSE score is rather small for each dataset.

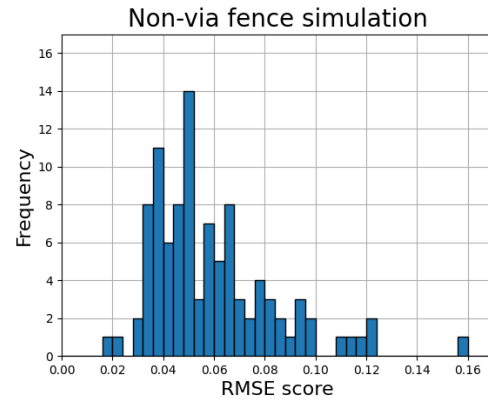
Table 5.3: Variance and standard deviation for predictions and simulations.

Analysis	Model	Variance	Standard deviation
Predicted	Non-via fence	0.00029	0.01708
	With via fence	0.00026	0.01626
Simulated	Non-via fence	0.00059	0.02430
	With via fence	0.00041	0.02019

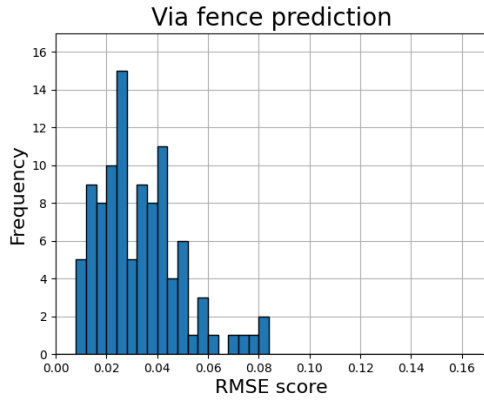
This indicates that the models are well generalized and not that sensitive to variations in circuit structure. The distribution of all RMSE scores for the 100 optimized circuits, are illustrated in Figure 5.3.



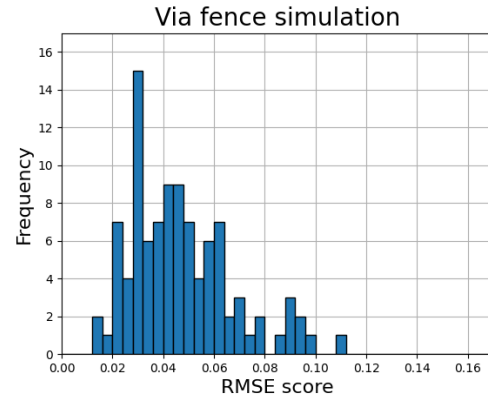
(a) Histogram for the predictions for the non-via fence case



(b) Histogram for the simulated RMSE for the non-via fence case



(c) Histogram for the predictions for the via fence case



(d) Histogram for the simulated RMSE for the via fence case

Figure 5.3: Histograms for the RMSE scores of the predictions and simulations from both models.

Notably the shape of the distribution for the simulated and predicted RMSE scores are rather similar, where both are right-skewed. These histograms are reasonable due to the model not obtaining perfect results.

In addition to comparing the S-parameters, a comparison of the generated circuits can also be made. The actual layout of each circuit is not important as the focus is the resulting S-parameters. However, it displays the two models' understanding of how each cell impacts the S-parameters. The circuits used to obtain the target S-parameters, and both generated circuits are illustrated in Figure 5.4, for the same random example as in Figure 5.1.

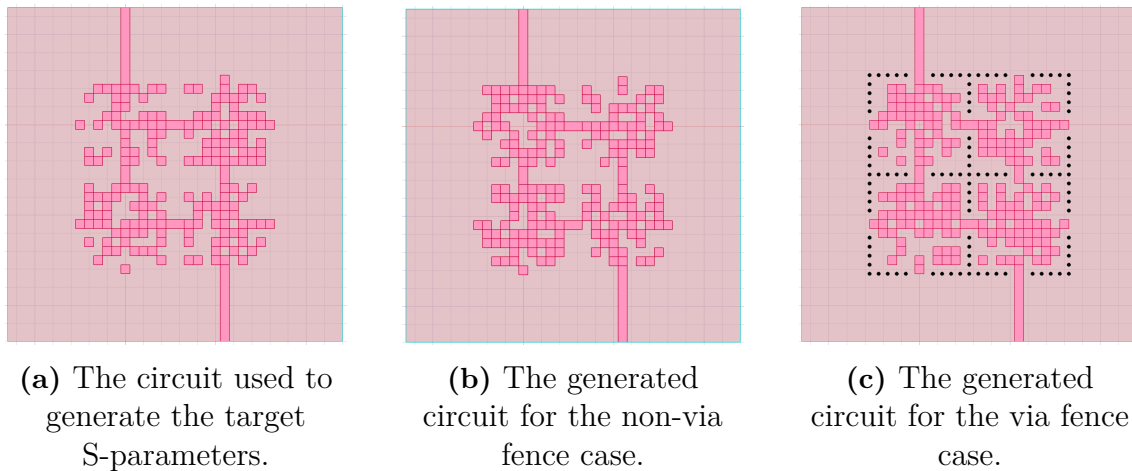
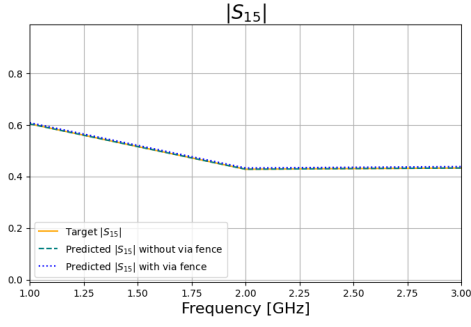


Figure 5.4: The modular circuit used to generate the target S-parameters, as well as the two generated circuits from the GA for both cases.

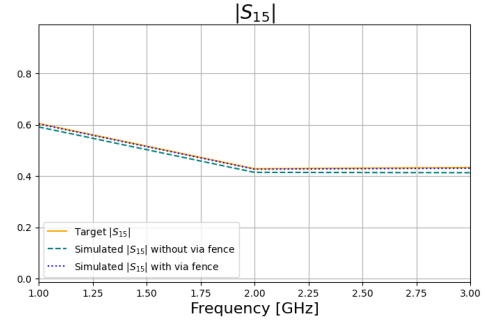
As can be seen in Figure 5.4a the initial circuit, constructed by circuits from the dataset, has a completely different design than the circuits created with genetic optimization. This is an expected result as the inverse design approach only cares about the resulting S-parameters, and not the structure of the circuit itself.

5.1.2 Frequency Analysis

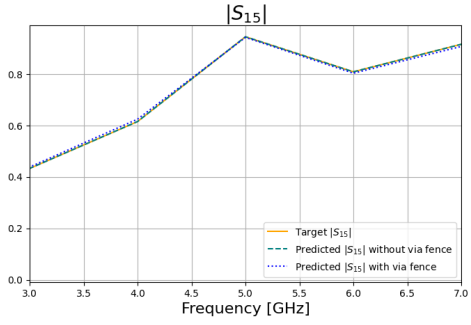
In Figure 5.2 it can be seen that the simulated value of the transmission S-parameters of both models vary slightly from the target S-parameters. In order to assess these variations the frequencies are divided into three spans; 1-3 GHz, 3-7 GHz and 7-10 GHz shown in Figure 5.5.



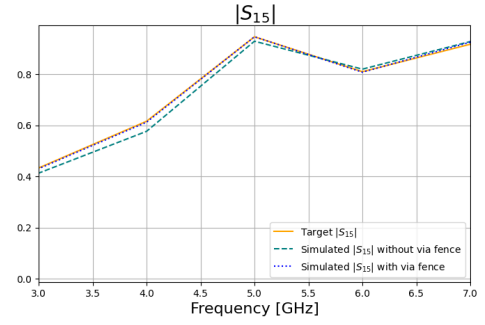
(a) $|S_{15}|$ for prediction and target for both cases in frequency range 1-3 GHz.



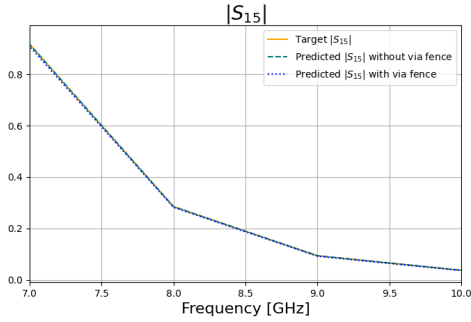
(b) $|S_{15}|$ for simulation and target for both cases in frequency range 1-3 GHz.



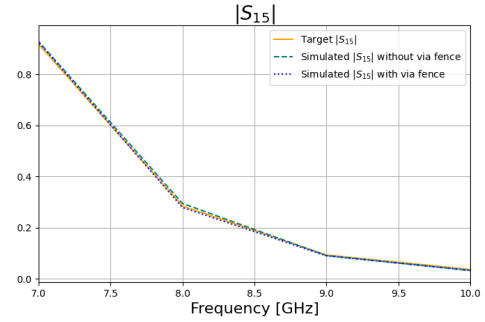
(c) $|S_{15}|$ for prediction and target for both cases in frequency range 3-7 GHz.



(d) $|S_{15}|$ for simulation and target for both cases in frequency range 3-7 GHz.



(e) $|S_{15}|$ for prediction and target for both cases in frequency range 7-10 GHz.



(f) $|S_{15}|$ for simulation and target for both cases in frequency range 7-10 GHz.

Figure 5.5: Plot of different frequency ranges for a random example.

Upon closer inspection of the different frequencies, it is notable that the RMSE differences are still rather small. The S-parameters from the via fence model almost follows the target parameters perfectly throughout the whole span in Figure 5.5. However, the largest discrepancies are shown in Figure 5.5d, i.e, the middle section of the frequency span. This issue seems to be the same for all 100 test circuits, as can be seen in Figure 5.6 and Figure 5.7.

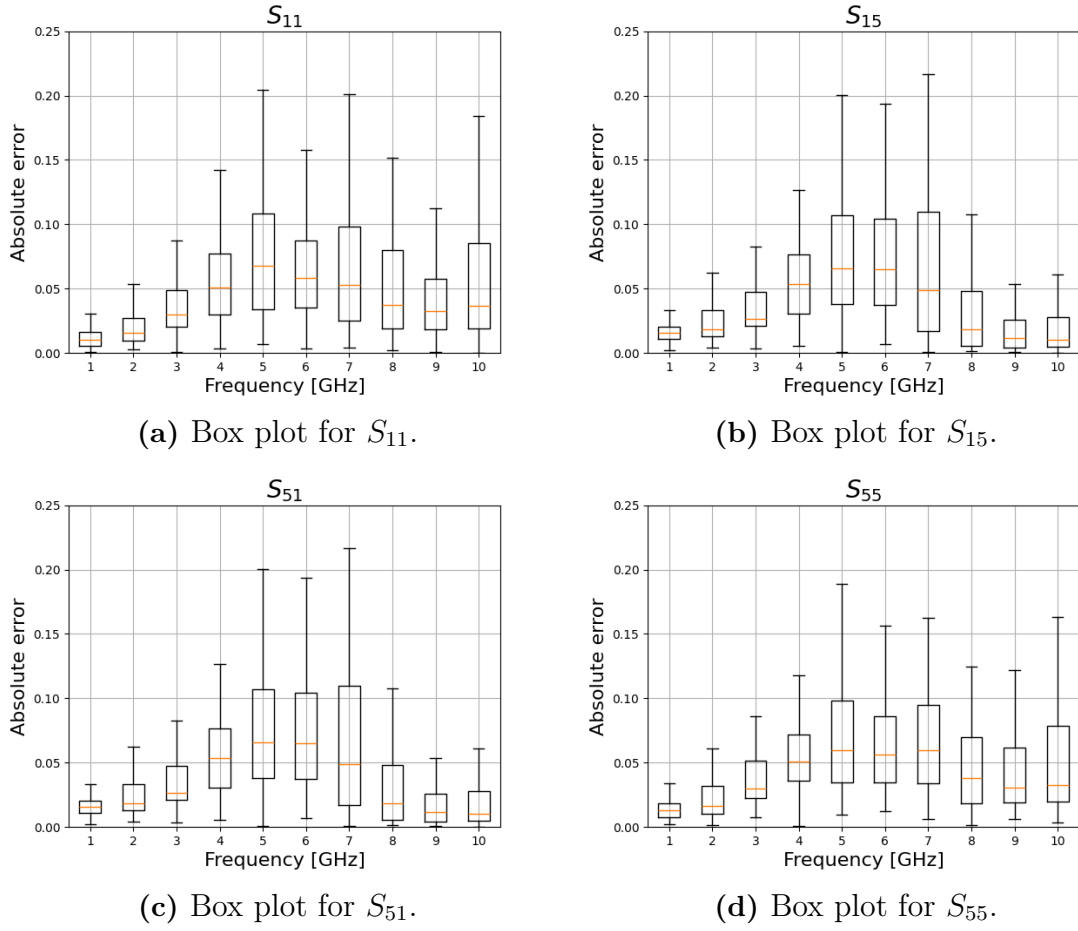


Figure 5.6: Box plots for the non-via fence model.

The frequency errors of the non via fence case seem to have the biggest outliers in the middle span for both reflection and transmission. Results for the lower frequencies are quite good for both reflection and transmission. In comparison the results for higher frequencies are rather good for transmission, but poor for reflection. If the median and the interquartile are compared to their respective counterparts for the via fence case, in Figure 5.7, it is possible to see an improvement over all frequencies for both transmission and reflection. This is sign that using a via fence can improve the robustness of the module-based design concept by minimizing the noise and interference over multiple frequencies.

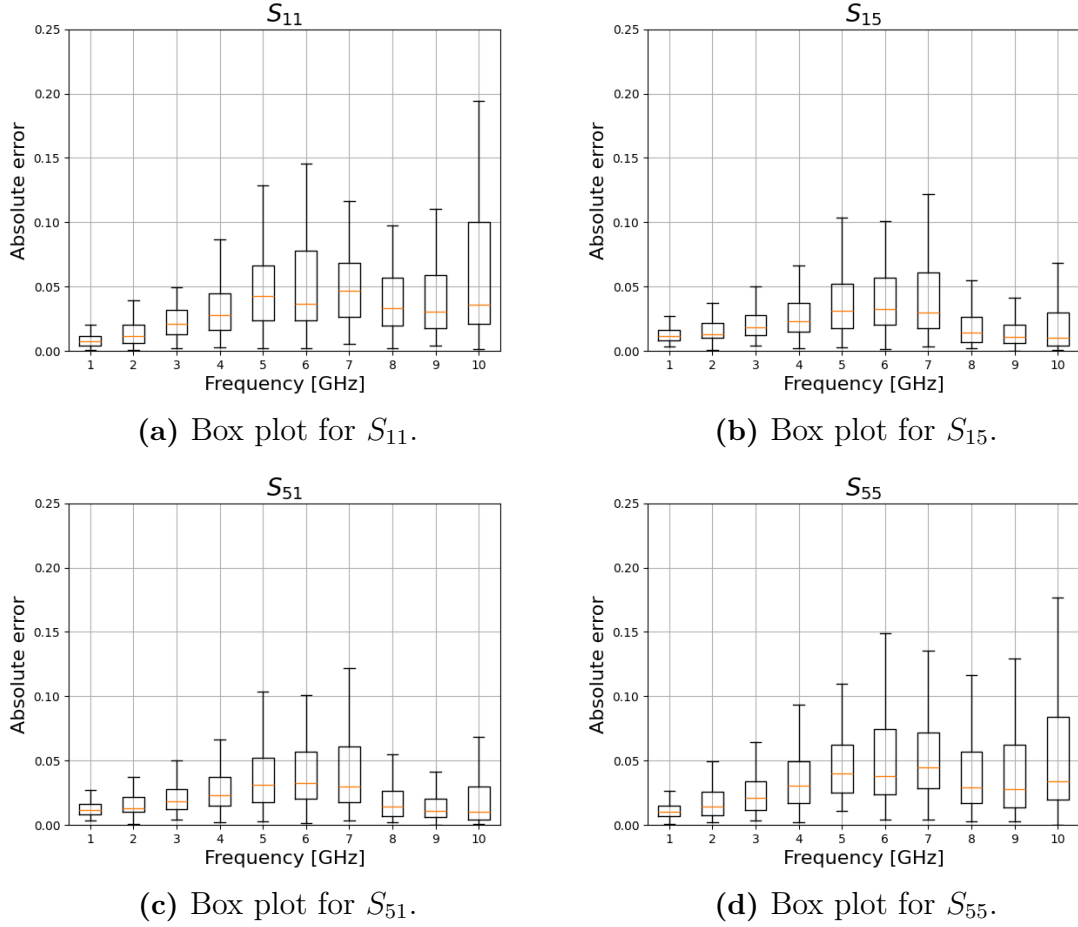


Figure 5.7: Box plots for the via fence model.

The reason that the lower frequencies have smaller errors compared to higher frequencies is due to the circuit dimensions being small relative to the wavelength, thus making the S-parameters easier to predict. In the subplots there are some contrary signs to this as the higher frequencies for transmission being better than the middle frequencies. Such results are hard to analyze fully as complete knowledge of the exploration during both training and optimization is unknown. The model might have found a way to lower the RMSE score during training and optimization by focusing on higher and lower frequencies, instead of evenly lowering the RMSE score for all frequencies. The training and optimization has always been aiming for a lower mean RMSE for all frequencies combined. Thus, some frequencies could technically be only partially addressed in the process. Most likely, these issues are a mixture of structure design issue combined with model behavior.

5.2 Time Efficiency

Time efficiency is one of the main reasons for examining the modular concept. As the size of the circuits increases, the simulation and optimization times increase. However, fully evaluating the modular concept would require an enormous amount

of simulations. Thus, the results in this section will focus on time consumption to get the results shown in section 5.1. Secondly, the focus will be put on the via-fences implications on the time consumption for different parts of the project. The hardware used is showcased in Table 5.4.

Table 5.4: Lists of hardware used.

Hardware	Project part
2x Intel(R) Xeon(R) Gold 6242	Simulations
Tesla V100	ML and GA

The most time consuming task in this thesis was the simulations of the two datasets. Simulating Maxwell’s equations for the circuits is time consuming for many reasons. The large number of circuits needed to be simulated means that a considerable amount of memory is needed. The circuits in the dataset thus have to be separated into smaller batches in order for the clusters to be able to simulate them. The amount of simulation hours that were spent, where losses such failed simulations are excluded, are shown in Table 5.5.

Table 5.5: Simulation time for each dataset.

Configuration	Number of Simulations	Simulation Time (hours)
Non-via fence	30 000	65.5
With via fence	30 000	397

Notably, the non-via fence simulations require much shorter simulation time than the via fence designs. One of many reasons for this is most likely due to the cylindrical form of the vias, as simulating the impact of using a cylinder is more time-consuming compared to rectangular shapes.

After each dataset was constructed the ML-model was trained, and the time consumption for the training is illustrated in Table 5.6.

Table 5.6: The time required for training the ML model for each dataset.

Configuration	Number of Epochs	Training Time (hours)
Non-via fence	245	1.2
With via fence	201	0.9

The difference between the ML-model training times for the different datasets is very small. This is logical as they are structurally closely comparable, with the main difference being the number of neurons used. This is seen as the lower complexity of the via fence model leads to a lower number of epochs, and a lower total time spent training. Furthermore, unlike for the simulations, the usage of a via fence does not have a negative impact on the ML training. This is due to the ML only being fed the circuit matrices and the corresponding S-parameters.

After the ML-models had been trained, the genetic algorithm was used to find multiple circuits. The GAs used in this thesis had large time-consumption in comparison to what was expected. The time required for 100 modular circuits with a limit of 100 iterations is shown in Table 5.7

Table 5.7: Time for the GA optimization to reach termination requirement for each dataset.

Configuration	Number of Iterations	Number of Circuits Generated	Full optimization Time (hours)
Non-via fence	100	100	8.3
With via fence	100	100	8.3

The main reason for the GA process taking a long time is due to the mathematical calculations that have to be done in each iteration. As the model is solely able to predict S-parameters for 9×9 matrices, the model has to do four predictions in each step. Then all four predictions have to go through the calculations in section 3.3. As the optimization process consists of 100 circuits, each having a limit of 100 iterations, the calculations have to be performed 10 000 times. Thus, a balance between time consumption and number of iterations must be found. In the test cases that were conducted for 5.7 the main focus was to achieve the best results. To battle this issue, an idea is experimenting with the termination requirements. An extra requirement could be added that evaluates whether the optimization has become stagnant over multiple iterations and terminate the optimization. Nonetheless, the time usage for the two cases are rather similar.

It is notable that the via fence dataset is more time consuming during the simulations compared to the non-via fence dataset. In spite of that, the via fence ML-model and its simulated circuit designs have better results. Consequently, the improved results can be weighed against the extra time consumption. One aspect which speaks

for the via fence dataset to be preferred over the non-via fence dataset is that the simulation only have to be ran once. As the simulations in the modular concept allows for multiple sized circuits, the same dataset can be used for any module topology. In cases where the modular concept is not applied, the via fence is not required as there will be no other modules that can cause interference. However, if more modules were to be used, the via fence would have a higher impact as more modules would lead to more interference.

6

Conclusion

The thesis has assessed the potential of a module-based concept for circuit design using machine learning in the 1-10 GHz range. In order to evaluate if the modular concept was feasible a ML-model as well as a genetic algorithm were constructed that could generate modular circuits based on targeted S-parameters. The modular concept shows good results, as the generated circuits from the genetic algorithm had low RMSE scores. The generated circuits were also simulated which resulted in slightly different S-parameters compared to the S-parameters of the GA. This was logical as the error in the ML-model's accuracy only manifests itself during the simulations. Due to this error, the error margin between the predicted S-parameters of the generated circuits and target S-parameters was small. However, it was noticeably higher when the circuits were simulated. There were also notable differences for different frequencies in the 1-10 GHz range, where the lower and higher frequencies performed better than the middle frequencies. This behavior was not expected, but one reason could be improper training. As the model was not trained for each frequency individually, the model might put less emphasis on certain frequencies.

Although the results are favourable, there are certain elements that limits the effectiveness of the modular concept. Mainly this is due to the genetic algorithm, as the increased computational complexity introduced by the calculations of the S-parameters of the full circuits reduces the time efficiency.

As for the impact of the via fence, it can be concluded that using a via fence has little to no impact on the accuracy of the ML-model in its prediction on a circuit's S-parameters. The via fence does not improve the prediction compared to the non-via fence model on it's ability to successfully generate a circuit given some S-parameter targets. The main effect of utilizing such a structure is seen when verifying the results in HFSS, as the via fence shields the modules from eachother, preventing interference and increasing performance. The accuracy of the S-parameters of the generated circuits using a via fence outperformed the non-via fence generated circuits, even though ML-model trained on the via fence dataset had a slightly worse performance. The downside however, is that simulating these circuits took a considerably longer time compared to the non-via fence circuits.

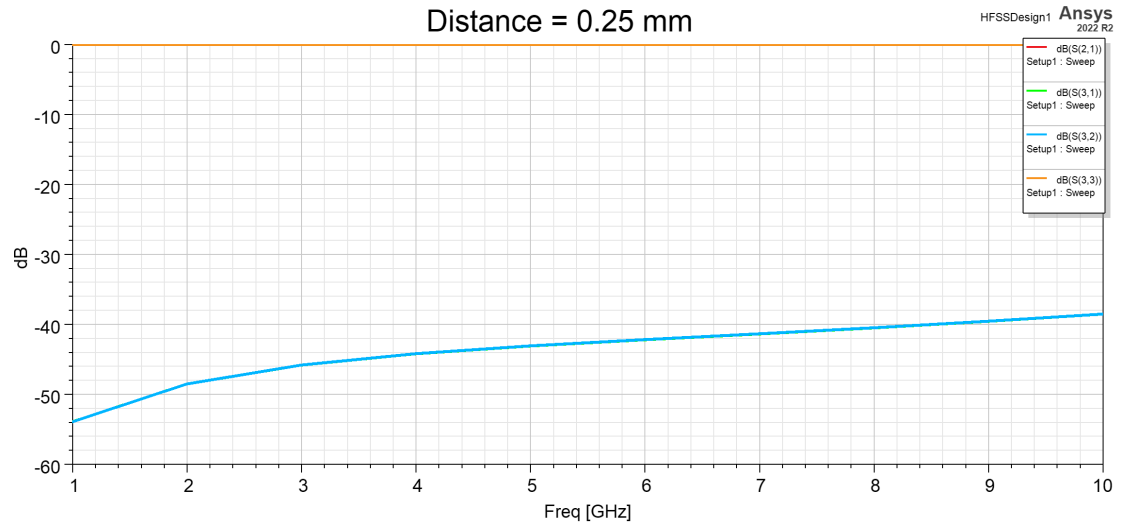
Bibliography

- [1] E. A. Karahan, Z. Liu, and K. Sengupta, “Deep-learning-based inverse-designed millimeter-wave passives and power amplifiers,” *IEEE Journal of Solid-State Circuits*, vol. 58, no. 11, pp. 3074–3088, 2023. DOI: 10.1109/JSSC.2023.3276315.
- [2] M. H. Tahersima, K. Kojima, T. Koike-Akino, *et al.*, “Deep neural network inverse design of integrated photonic power splitters,” *Scientific reports*, vol. 9, no. 1, p. 1368, 2019, ISSN: 2045-2322. DOI: 10.1038/s41598-018-37952-2. [Online]. Available: <https://europepmc.org/articles/PMC6361971>.
- [3] D. Lee, G. Shin, S. Lee, K. Kim, T.-H. Oh, and H.-J. Song, “Neural-network-based automated synthesis of transformer matching circuits for rf amplifier design,” *IEEE Transactions on Microwave Theory and Techniques*, vol. PP, pp. 1–14, Nov. 2022. DOI: 10.1109/TMTT.2022.3199756.
- [4] “Optimizing 1d dielectric electromagnetic bandgap (d-ebg) structures using multistage genetic algorithm (ms-ga) and considering parameter variations,” *Progress In Electromagnetics Research B*, vol. 103, pp. 1–18, 2023. DOI: 10.2528/PIERB23071001.
- [5] S. Ersoy and T. Waqar, “Surface acoustic wave sensing,” in *Encyclopedia of Sensors and Biosensors (First Edition)*, R. Narayan, Ed., First Edition, Oxford: Elsevier, 2023, pp. 129–141, ISBN: 978-0-12-822549-3. DOI: <https://doi.org/10.1016/B978-0-12-822548-6.00103-5>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780128225486001035>.
- [6] A. Ferrero and M. Pirola, “Generalized mixed-mode s-parameters,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 1, pp. 458–463, 2006. DOI: 10.1109/TMTT.2005.860497.
- [7] R. E. Collin, *Foundations for microwave engineering*. (IEEE press series on electromagnetic wave theory). IEEE Press, 2001, ISBN: 0780360311. [Online]. Available: <https://ieeexplore.ieee.org/servlet/opac?bknumber=5265446>.
- [8] A. Suntives, A. Khajooeizadeh, and R. Abhari, “Using via fences for crosstalk reduction in pcb circuits,” in *2006 IEEE International Symposium on Electromagnetic Compatibility, 2006. EMC 2006.*, vol. 1, 2006, pp. 34–37. DOI: 10.1109/ISEMC.2006.1706258.
- [9] T. Mitchell, *Machine Learning* (McGraw-Hill International Editions). McGraw-Hill, 1997, ISBN: 9780071154673. [Online]. Available: <https://www.cs.cmu.edu/~tom/mlbook.html>.
- [10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: <http://www.deeplearningbook.org>.

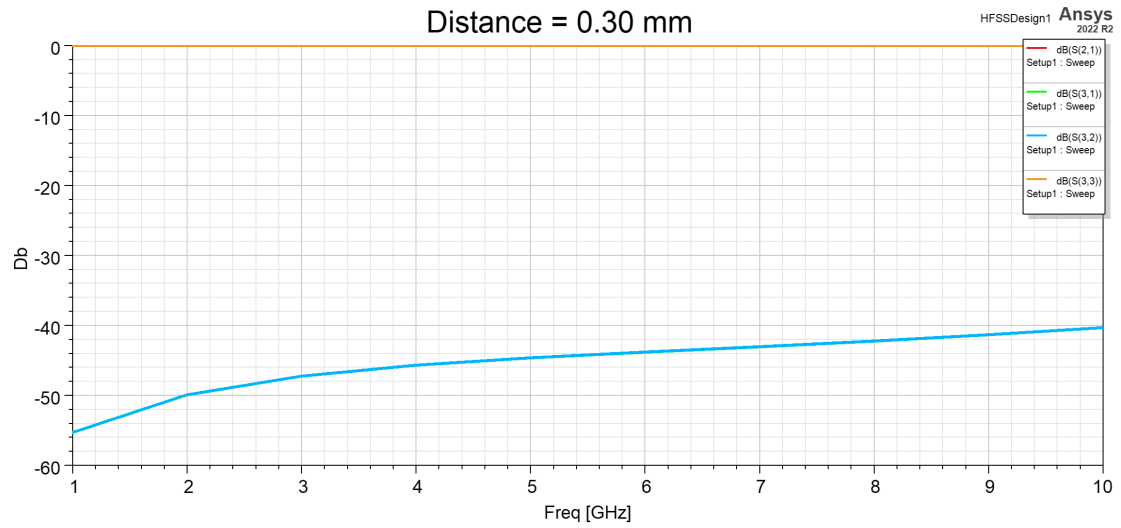
- [11] R. Pugliese, S. Regondi, and R. Marini, "Machine learning-based approach: Global trends, research directions, and regulatory standpoints," *Data Science and Management*, vol. 4, pp. 19–29, 2021, ISSN: 2666-7649. DOI: <https://doi.org/10.1016/j.dsm.2021.12.002>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666764921000485>.
- [12] S. Indolia, A. K. Goswami, S. Mishra, and P. Asopa, "Conceptual understanding of convolutional neural network- a deep learning approach," *Procedia Computer Science*, vol. 132, pp. 679–688, 2018, International Conference on Computational Intelligence and Data Science, ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2018.05.069>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050918308019>.
- [13] A. Laith, Z. Jinglan, and et al., "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, 2021. DOI: <https://doi.org/10.1186/s40537-021-00444-8>. [Online]. Available: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8>.
- [14] J. Patterson and A. Gibson, *Deep learning : a practitioner's approach*. O'Reilly, 2017, ISBN: 9781491914250.
- [15] R. Pal, "Chapter 4 - validation methodologies," in *Predictive Modeling of Drug Sensitivity*, R. Pal, Ed., Academic Press, 2017, pp. 83–107, ISBN: 978-0-12-805274-7. DOI: <https://doi.org/10.1016/B978-0-12-805274-7.00004-X>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B978012805274700004X>.
- [16] C. Aliferis and G. Simon, "Overfitting, underfitting and general model overconfidence and under-performance pitfalls and best practices in machine learning and ai," in *Artificial Intelligence and Machine Learning in Health Care and Medical Sciences: Best Practices and Pitfalls*. Springer International Publishing, 2024, pp. 477–524, ISBN: 978-3-031-39355-6. DOI: 10.1007/978-3-031-39355-6_10. [Online]. Available: https://doi.org/10.1007/978-3-031-39355-6_10.
- [17] K. Sastry, D. Goldberg, and G. Kendall, "Genetic algorithms," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer US, 2005, pp. 97–125, ISBN: 978-0-387-28356-2. DOI: 10.1007/0-387-28356-0_4. [Online]. Available: https://doi.org/10.1007/0-387-28356-0_4.
- [18] J. McCall, "Genetic algorithms for modelling and optimisation," *Journal of Computational and Applied Mathematics*, vol. 184, no. 1, pp. 205–222, 2005, Special Issue on Mathematics Applied to Immunology, ISSN: 0377-0427. DOI: <https://doi.org/10.1016/j.cam.2004.07.034>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377042705000774>.
- [19] L. Zhang, H. Chang, and R. Xu, "Equal-width partitioning roulette wheel selection in genetic algorithm," in *2012 Conference on Technologies and Applications of Artificial Intelligence*, 2012, pp. 62–67. DOI: 10.1109/TAAI.2012.21.
- [20] P. Kora and P. Yadlapalli, "Crossover operators in genetic algorithms: A review," *International Journal of Computer Applications*, vol. 162, pp. 34–36, Mar. 2017. DOI: 10.5120/ijca2017913370.

A

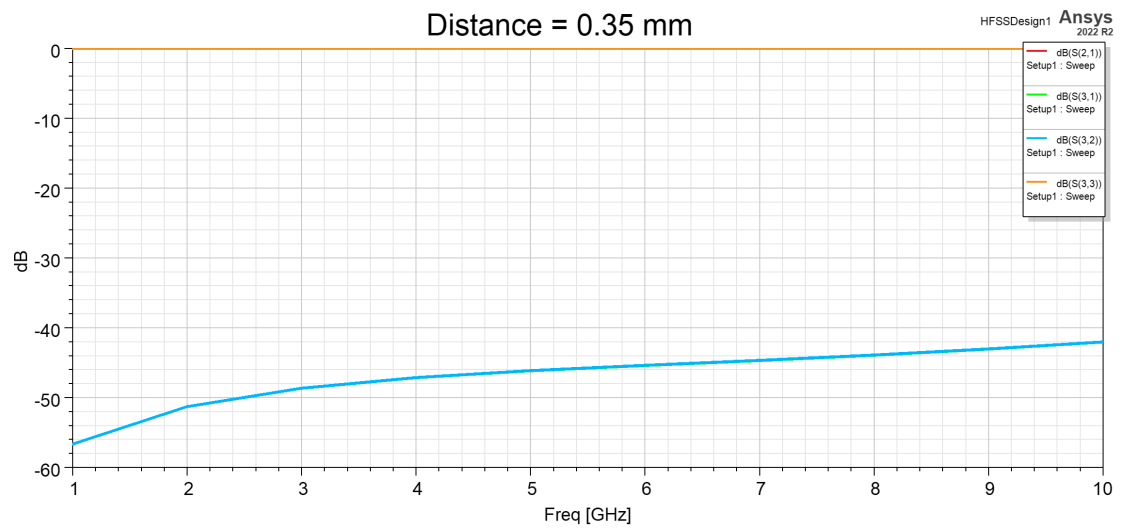
Appendix 1



(a) Distance = 0.25 mm.

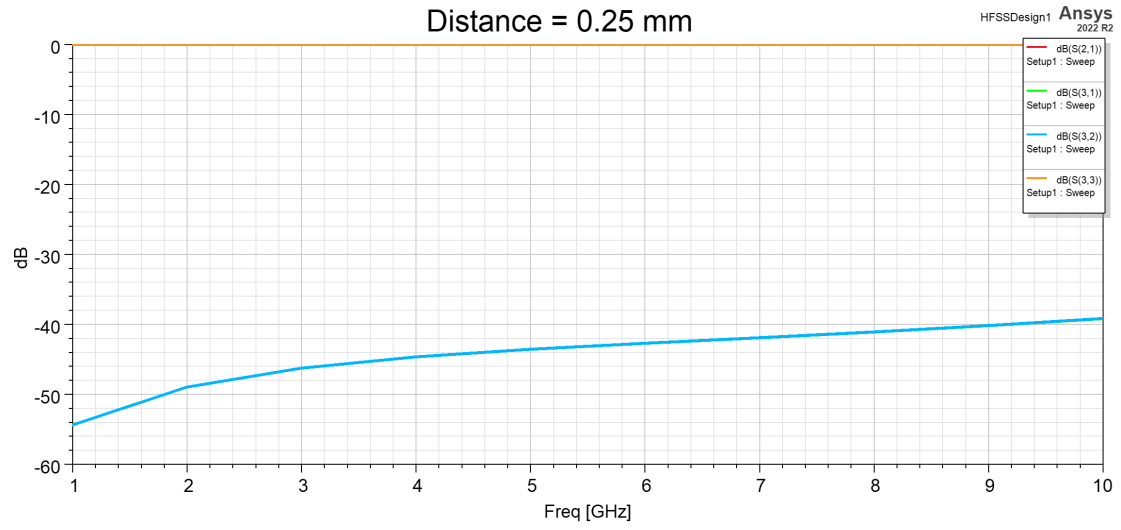


(b) Distance = 0.30 mm.

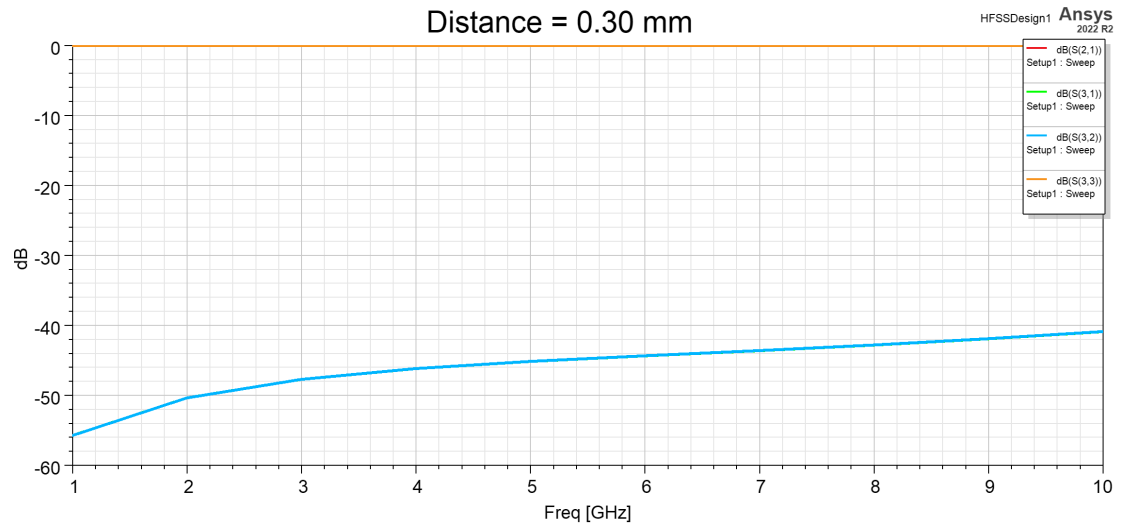


(c) Distance = 0.35 mm.

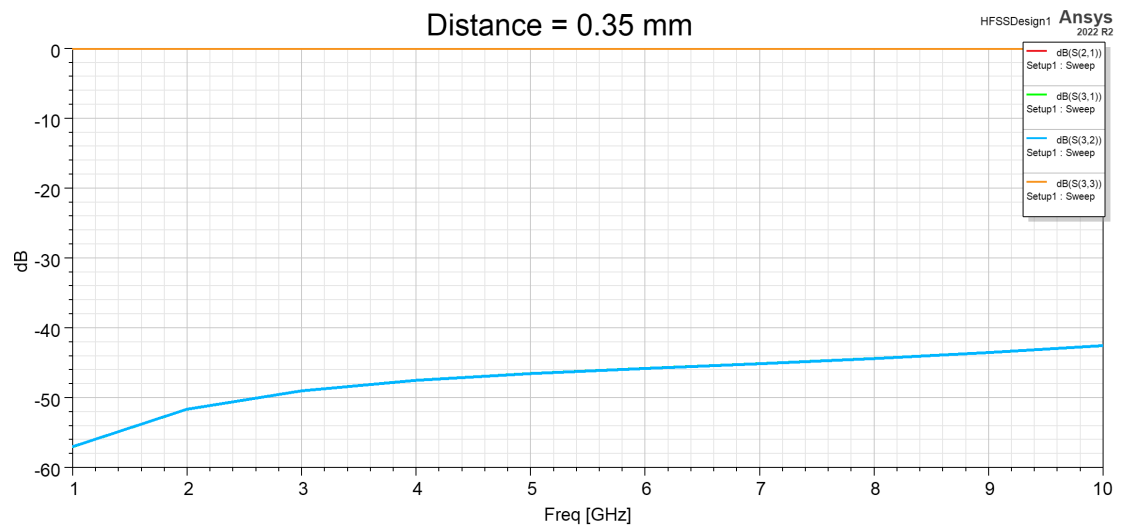
Figure A.1: The result from the simulation with a cylinder diameter of 0.150 mm.



(a) Distance = 0.25 mm.

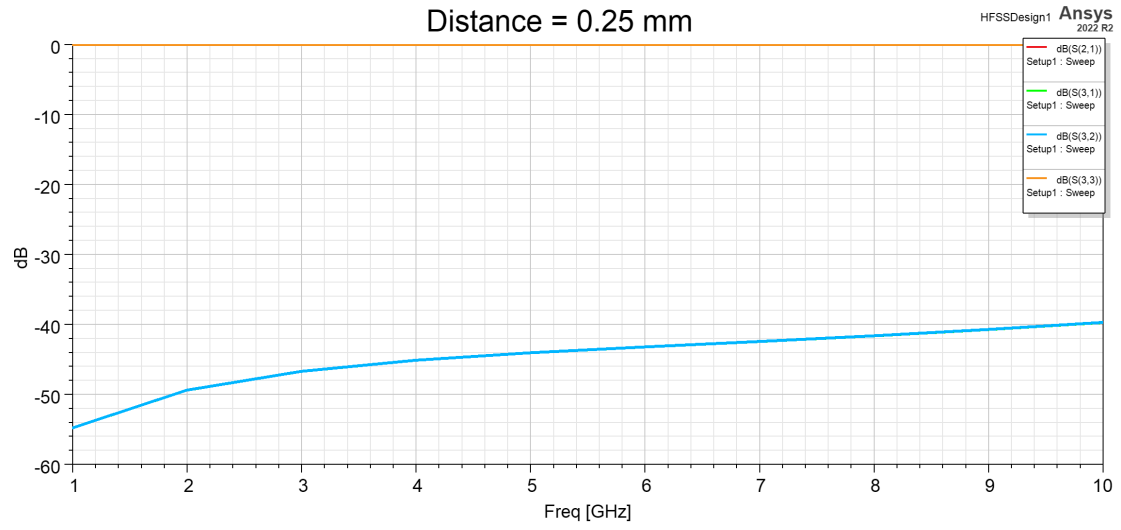


(b) Distance = 0.30 mm.

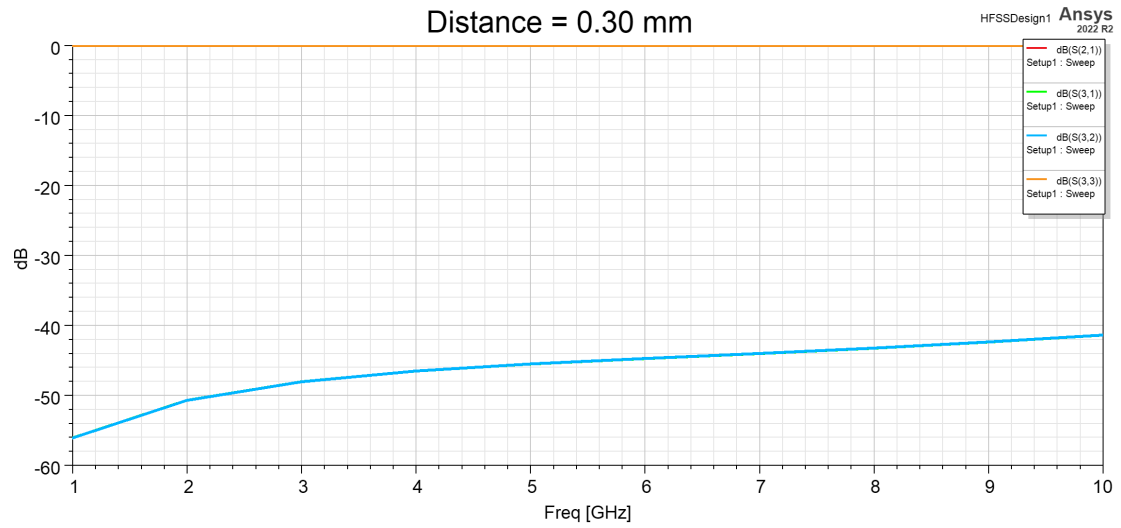


(c) Distance = 0.35 mm.

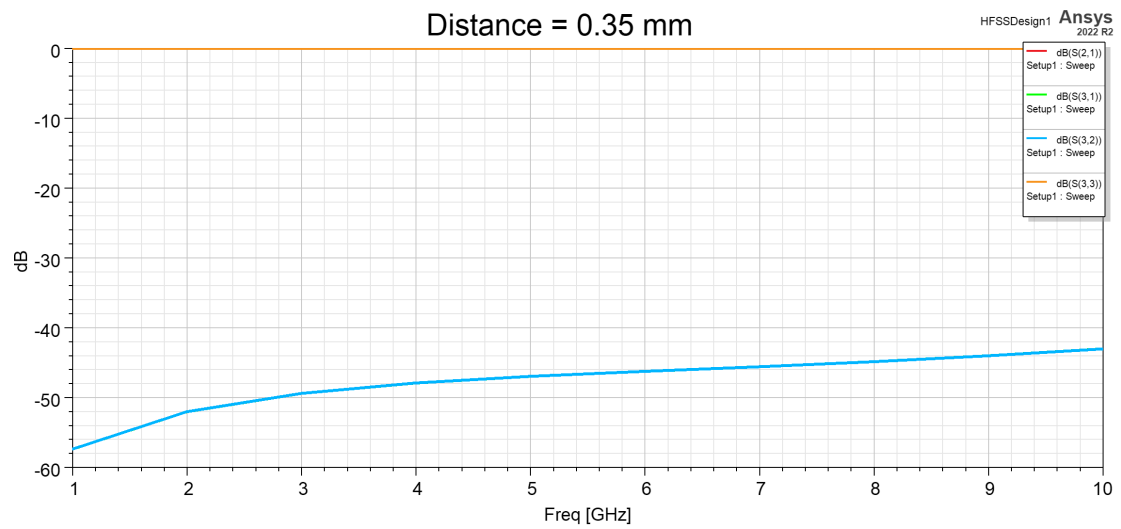
Figure A.2: The result from the simulation with a cylinder diameter of 0.175 mm.



(a) Distance = 0.25 mm.

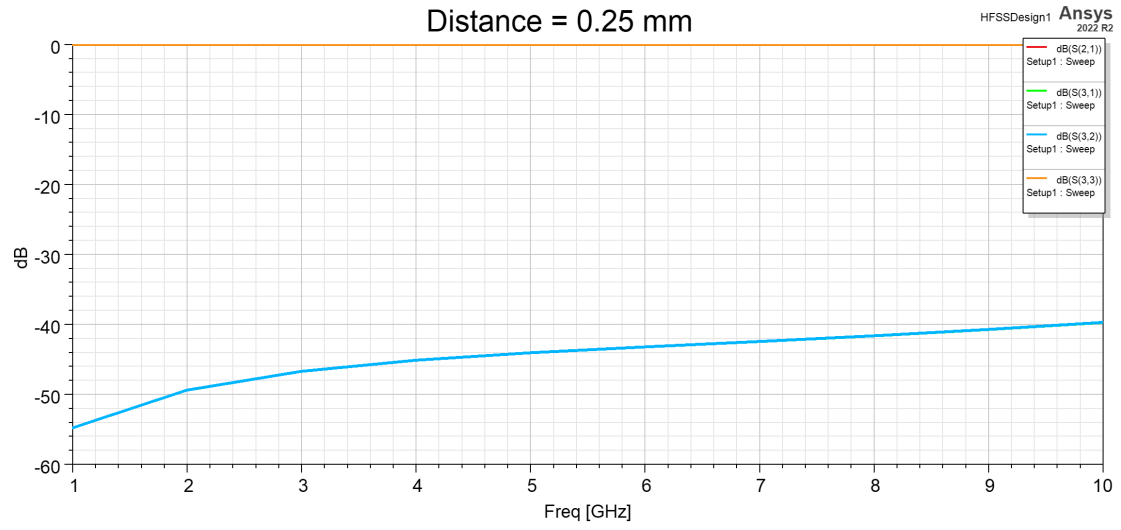


(b) Distance = 0.30 mm.

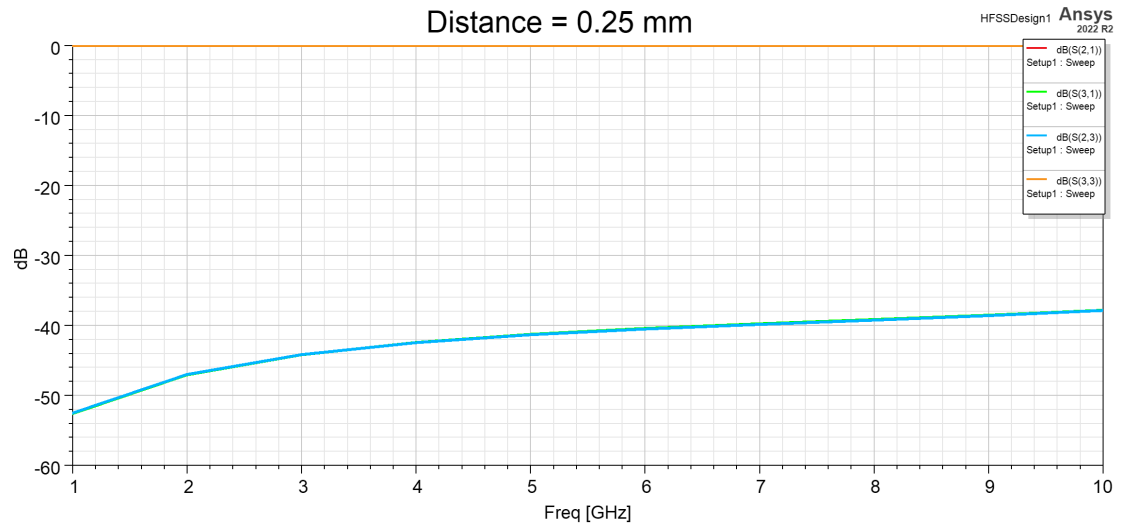


(c) Distance = 0.35 mm.

Figure A.3: The result from the simulation with a cylinder diameter of 0.200 mm.



(a) The result with a via fence.



(b) The result without a via fence.

Figure A.4: Comparison of EMI on two circuits.

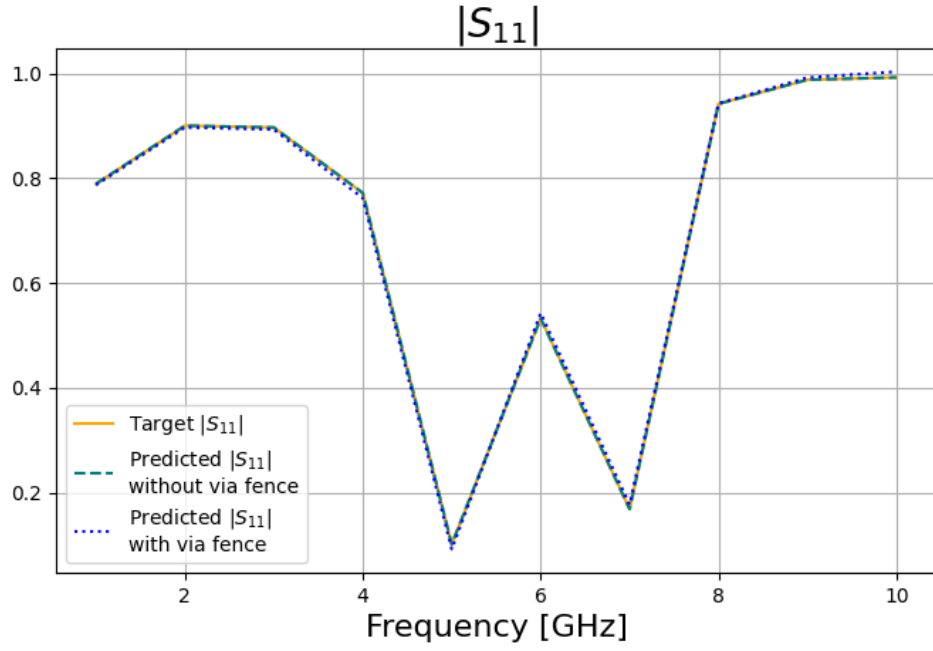


Figure A.5: Plot of $|S_{11}|$ for prediction and target for both models.

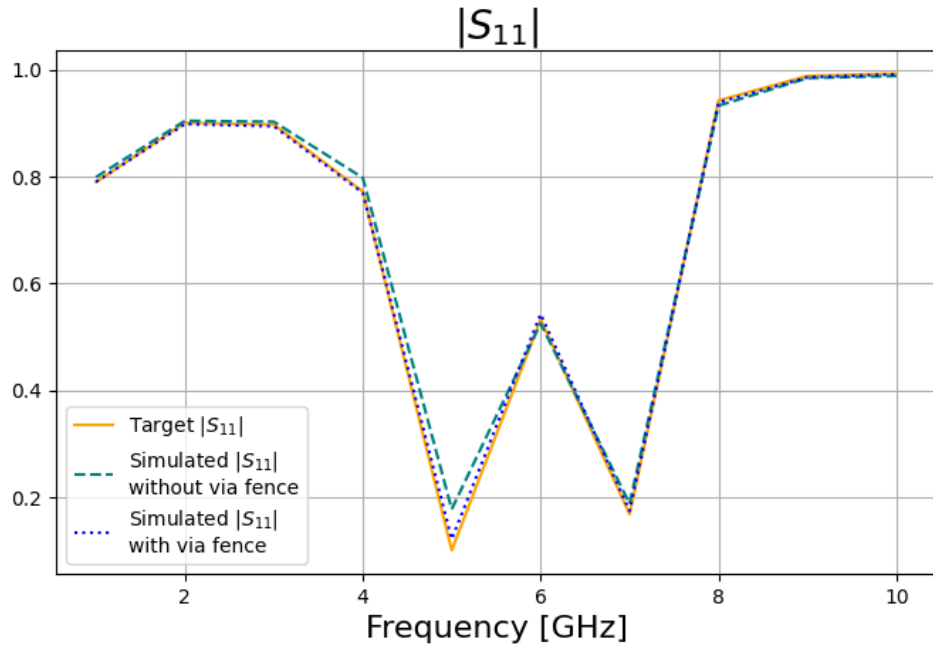


Figure A.6: Plot of $|S_{11}|$ for the simulation and target for both models.

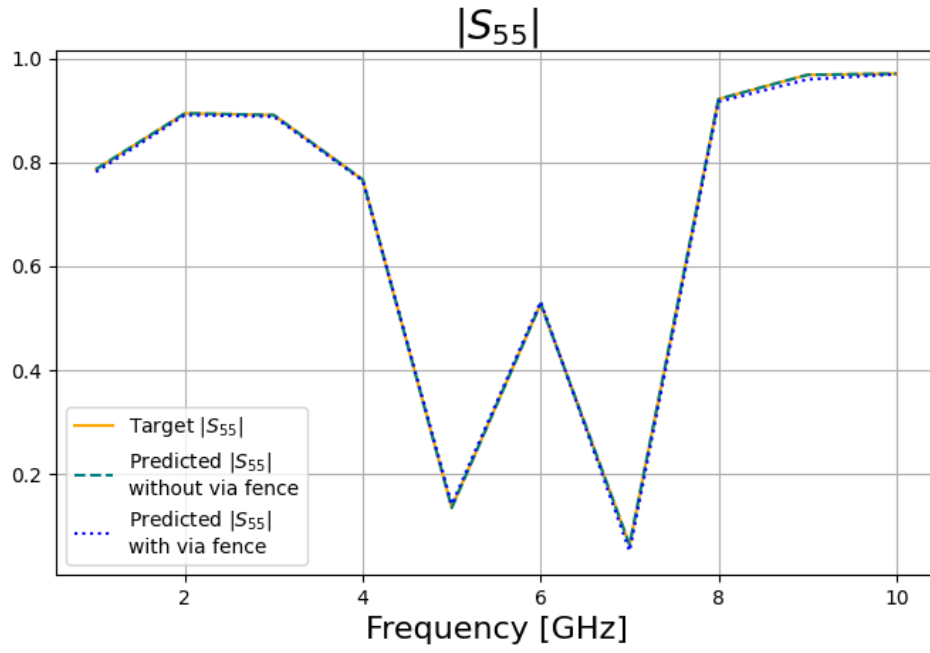


Figure A.7: Plot of $|S_{55}|$ for prediction and target for both models.

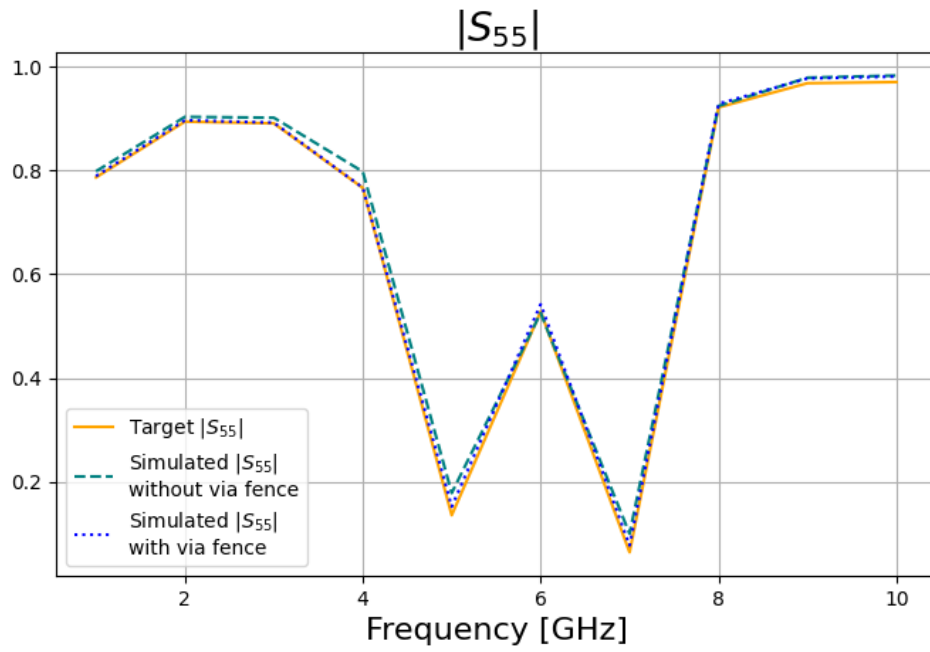


Figure A.8: Plot of $|S_{55}|$ for simulation and target for both models.

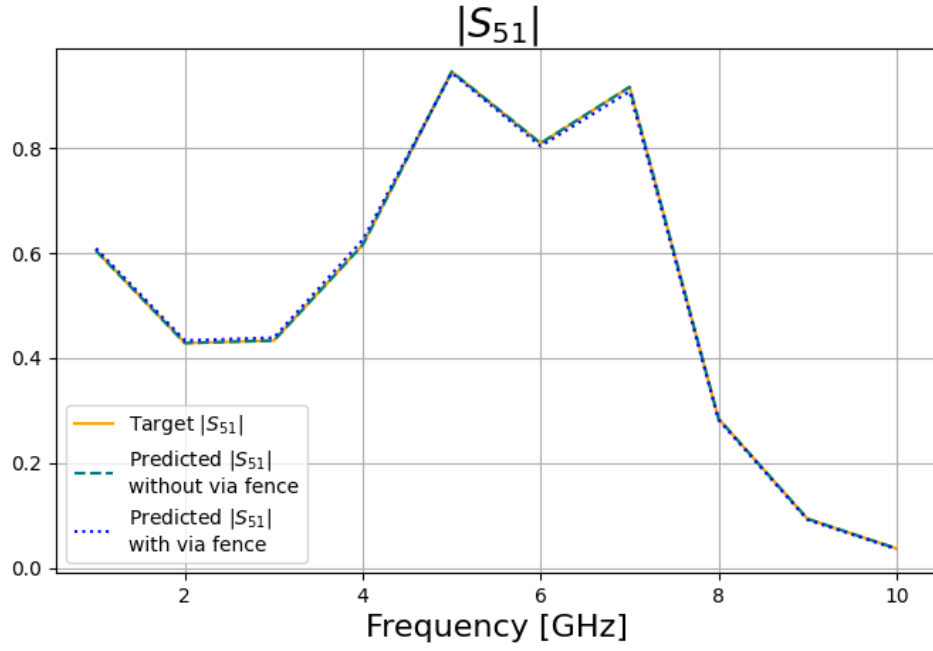


Figure A.9: Plot of $|S_{51}|$ for prediction and target for both models.

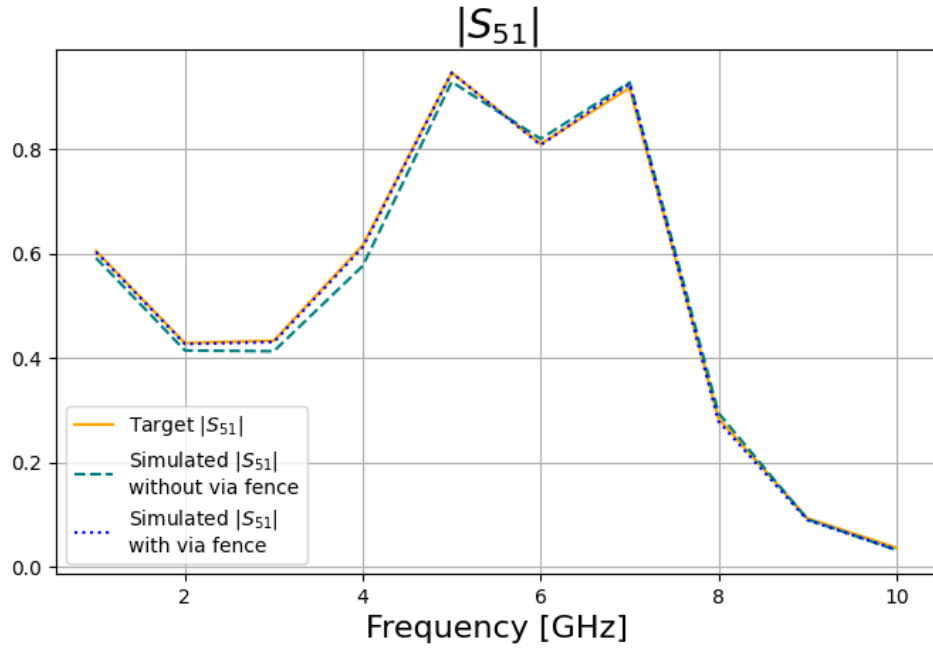


Figure A.10: Plot of $|S_{51}|$ for simulation and target for both models.

DEPARTMENT OF MICROTECHNOLOGY AND NANOSCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se



CHALMERS
UNIVERSITY OF TECHNOLOGY