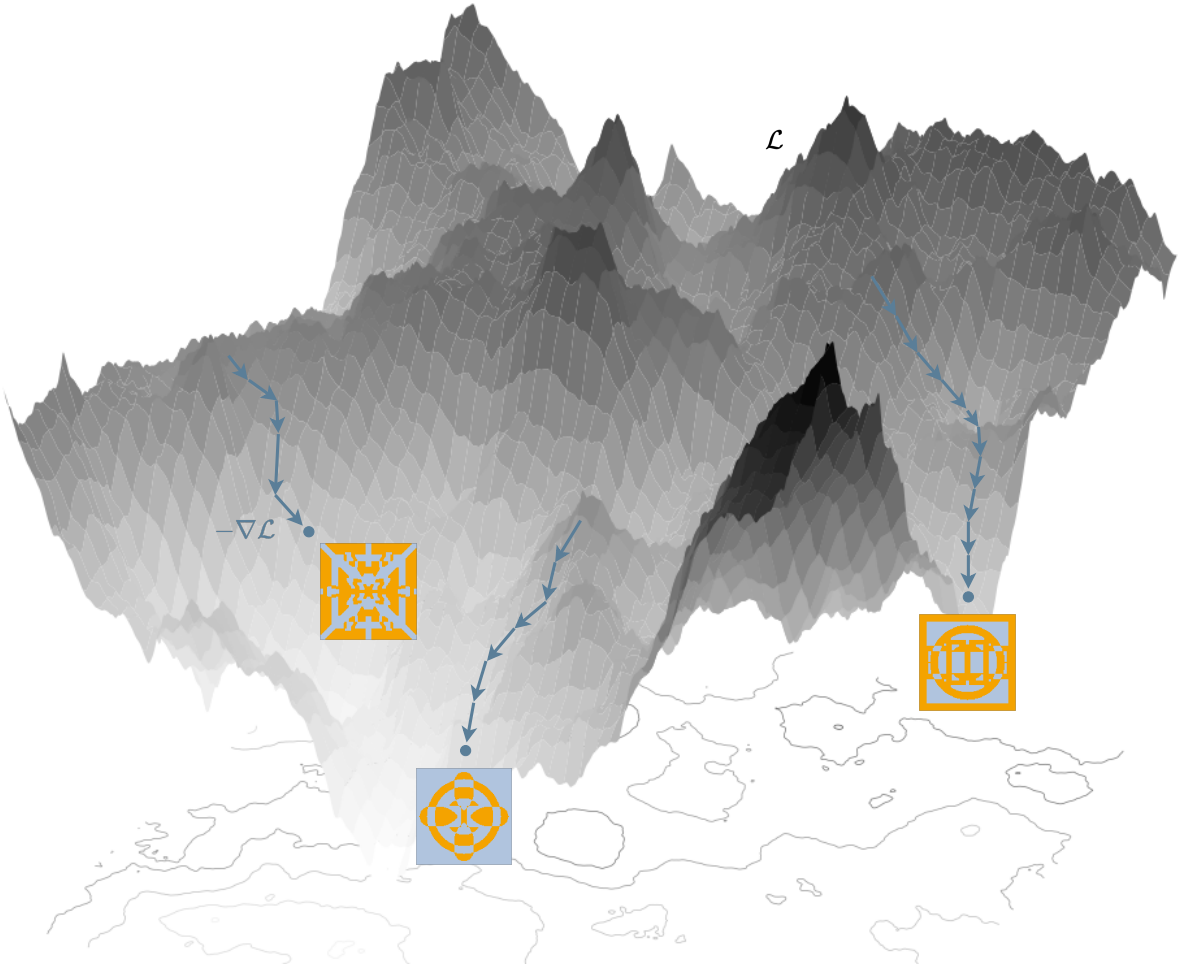
CHALMERS
UNIVERSITY OF TECHNOLOGY



From Noise to Pattern: Inverse Design of FSS Using Variational Autoencoder

Generative Machine Learning for the Inverse Design of FSS

Master's thesis in Complex Adaptive Systems

FRANCISCO BOUDAGH

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

From Noise to Pattern: Inverse Design of FSS Using Variational Autoencoder

Generative Machine Learning for the Inverse Design of FSS

FRANCISCO BOUDAGH

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2025

From Noise to Pattern: Inverse Design of FSS Using Variational Autoencoder Generative Machine Learning for the Inverse Design of FSS
FRANCISCO BOUDAGH

Supervisor: Ahmed Gouda, Department of Antenna System and Technology, Ericsson AB
Supervisor: Giovanni Volpe, Department of Physics, University of Gothenburg

Examiner: Giovanni Volpe, Department of Physics, University of Gothenburg

Master's Thesis 2025
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Conceptual visualization of the high-dimensional loss landscape in FSS design, where every point corresponds to a unique FSS pattern. The arrows trace gradient descent paths, guiding the way to lower loss and optimal solutions.

Typeset in LaTeX
Gothenburg, Sweden 2025

From Noise to Pattern: Inverse Design of FSS Using Variational Autoencoder Generative Machine Learning for the Inverse Design of FSS

FRANCISCO BOUDAGH
Department of Physics
Chalmers University of Technology

Abstract

Frequency Selective Surfaces (FSSs) are critical for modern electromagnetic filtering, but their design traditionally relies on costly trial-and-error simulations. We present a generative machine learning framework for the inverse design of FSS unit cell patterns. By utilizing a conditional variational autoencoder (cVAE), the proposed method directly maps desired electromagnetic scattering parameters (S-parameters) to FSS patterns, thereby circumventing the traditional energy- and time-expensive trial-and-error approach inherent in FSS design. A dataset of 10,000 simulated (pattern, S-parameter) samples was generated using Ansys HFSS over the 2 to 8 GHz frequency range to train both a surrogate neural network, which accurately predicts the S-parameters from a given pattern, and a cVAE-based generator that synthesizes novel pattern designs conditioned on target frequency responses.

The integrated framework employs a gradient-based optimization strategy in the latent space to minimize the deviation between the predicted and desired S-parameter responses, with particular emphasis on preserving the resonant frequency. Benchmarking on the test dataset demonstrates that the surrogate model achieves mean absolute errors from 0.5 dB at 2 GHz to 1.9 dB at 8 GHz, while the optimization loop refines designs to yield deviations as low as 0.0-0.2 GHz at the resonant frequency for half of the samples. These results underscore the promising potential of generative machine learning for rapid FSS inverse design.

Keywords: Machine learning, variational autoencoder, artificial neural networks, inverse problem, optimization, frequency selective surfaces, metamaterials.

Acknowledgments

This thesis is the product of numerous conversations, interactions, and shared experiences. I am grateful to everyone who inspired me and encouraged authenticity throughout this journey, shaping both this work and my personal perspective.

Special thanks go to my supervisor, Ahmed, and my examiner, Giovanni, for their valuable guidance and insightful support.

To all who have supported me throughout my academic and personal journey, your influence is woven into every page of this thesis.

Francisco Boudagh, Gothenburg, June 2025

List of Acronyms

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

Adam	Adaptive Moment Estimation
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
cVAE	Conditional Variational Autoencoder
EM	Electromagnetic
FCNN	Fully Connected Neural Network
FSS	Frequency Selective Surface
HFSS	High-Frequency Structure Simulator
IRS	Intelligent Reflecting Surface
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
PEC	Perfect Electric Conductor
S-parameters	Scattering Parameters

Nomenclature

This nomenclature lists the parameters and variables used in the Methods, Results, and Conclusion chapters, excluding those exclusive to the Theory chapter.

Scattering Parameters

$\lvert S_{11} \rvert$	Magnitude of reflected waves (in dB)
$\lvert S_{21} \rvert$	Magnitude of transmitted waves (in dB)
$\mathbf{S}$	S-parameters vector (concatenation of $\lvert S_{11} \rvert$ and $\lvert S_{21} \rvert$)
$\hat{\mathbf{S}}$	Predicted S-parameters vector (obtained from the surrogate model)
$\mathbf{S}_t$	Target S-parameters vector
$\square_v$	Index at the valley of $\mathbf{S}_t$ (resonant frequency index)

Pattern Representation and Latent Space

$\mathcal{P}$	Pattern space
P	Pattern (binary image)
P^*	Optimal pattern derived from z^*
$(P, \mathbf{S})$	A pair consisting of a pattern and its corresponding S-parameters
$\mathcal{Z}$	Latent space of the variational autoencoder
z	A sample from the latent space of the variational autoencoder
z^*	Optimal latent space sample, obtained through optimization

Neural Network Models and Loss Functions

$\mathcal{N}_S$	Surrogate neural network model
$\mathcal{N}_G$	Generator neural network model
$\mathcal{L}_{\text{opt}}$	Loss function for the optimization process
$\mathcal{L}_S$	Loss function for training the surrogate neural network
$\mathcal{L}_G$	Loss function for training the generator neural network (VAE loss)

Electromagnetic Parameters

ε_r	Dielectric constant
$\tan\delta$	Dissipation factor
f	Frequency
λ	Wavelength
Δf	Frequency deviation
ΔMag.	Magnitude deviation

Contents

List of Figures

1

Introduction

1.1 Background

A Frequency Selective Surface (FSS) is an ultra-thin periodic surface designed to interact with incident electromagnetic plane waves by partly or fully transmitting, reflecting, or changing their polarization at different frequencies [1]. The smallest component of such a surface is called a unit cell, which is repeated along both axes, in theory infinitely and in practice many times depending on the specific application [1]. The common design consists of a dielectric substrate, serving as a complementary base, with a metal printed on it. The electromagnetic (EM) behavior of this surface depends, among other factors, on the pattern formed by the dielectric and metal [2]. Figure 1.1 depicts a simple FSS with a metallic crosshair-like pattern.

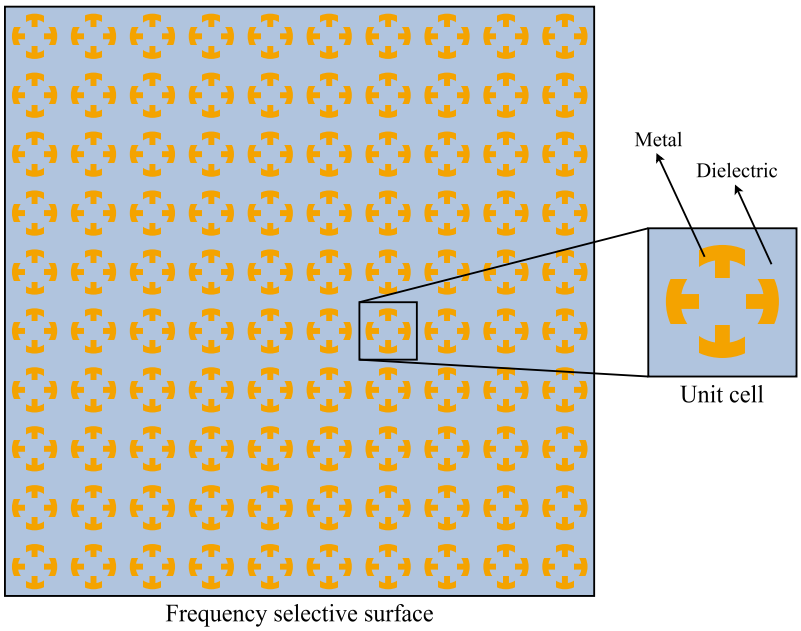


Figure 1.1: Example of an FSS with a unit cell that has a simple metallic crosshair-like pattern.

When designing an FSS, the pattern of a single unit cell is primarily considered. Other properties that affect the electromagnetic behavior include the specific mate-

rial choice (i.e., which dielectric and which metal), the number of FSS layers, and the geometric dimensions. When analyzing a specific design of an FSS, the scattering parameters (S-parameters) are typically used to quantify the EM behavior. In older literature, these are simply referred to as reflection and transmission coefficients. These parameters are complex numbers that describe the portion of the wave being transmitted or reflected at different frequencies. Thus, when designing the pattern, one aims to achieve the desired S-parameters in simulation software [1, 3].

The search for what is known today as an FSS began in the early 1950s by the organization now known as the Air Force Research Laboratory, operating under the United States Air Force. The main goal at that time was to minimize the radar signature of aircraft by using radar absorbing materials (RAM) [1], now referred to as stealth materials [4]. Today, FSS are used in many applications, including radar and stealth technology, satellite communication systems, wireless communication, medical applications, and optical and infrared applications [2]; specific applications will be presented later in this chapter.

1.2 Problem Statement and Thesis Scope

The current design process for FSSs typically begins with an experienced engineer or antenna designer defining the desired S-parameters, which may also be provided by the customer. Next, an initial metal-dielectric pattern is proposed and simulated. The resulting EM behavior is analyzed and compared with the target behavior. If the simulated and target behavior do not match, the geometric dimensions of the pattern are adjusted to better match the desired EM response. For example, if the proposed pattern is ring-shaped, the geometric dimensions would include the inner and outer radii. If these modifications are insufficient, an outer loop is initiated in which an entirely different pattern is selected that may better achieve the intended response [5].

The mapping between EM behavior and the FSS pattern is nonlinear and many-to-one; that is, multiple distinct patterns can produce the same EM response. In other words, the problem lacks a closed-form solution and is non-injective [6]. Consequently, the design process is trial-and-error-based, time-consuming, and energy-intensive due to the large number of required simulations. Furthermore, it relies heavily on the designer's experience and creativity [7].

To tackle the time- and energy-consuming, tedious process of designing FSS and to potentially discover new patterns for desired behavior, this thesis will develop a generative machine learning (ML) model that, given a desired electromagnetic behavior, comes up with the FSS pattern that obeys it. The idea is to use a conditioned variational autoencoder model as the pattern generator. There are many benefits and a need for such a tool. Given that such a model will generate reasonable or promising results, it will save enormous time, and consequently energy, and also lower the threshold for people with domain knowledge such that a junior designer will be able to design FSS. Furthermore, such a probabilistic model is, in contrast to

human experience and imagination, not limited and will come up with completely new and novel patterns.

There are other parameters besides the pattern that could affect the S-parameters, such as the dielectric substrate's physical properties, thickness, and the dimensions of the FSS itself, including height and width. All these parameters will be fixed. While it is possible to design multi-layer FSS to achieve more complicated EM behavior, to further limit the project's scope, only single-layer FSS will be used. The dielectric substrate will be fixed. Lastly, the metal type will also be fixed to perfect electric conductor (PEC). In conclusion, everything except for the unit cell pattern will be fixed.

1.3 Applications of FSS

Reducing the radar cross-section of electronic devices using antennas is one of the most common applications of FSS. This technique is widely used in the military to cloak stealth missiles or vehicles. For example, in a stealth missile such as the Franco-British Storm Shadow or the US F-35, an antenna in the nose is covered by a stealth radome consisting of FSS. This radome allows radar waves to pass through at a specific operating frequency (in band) while blocking waves at other frequencies (out of band) [2] (see Figure 1.3A).

A notable application of FSS worth highlighting is its use in advanced cloaking, a technique to manipulate the electromagnetic field grid in order to warp the incoming plane wave in a controlled manner to cloak an object (see Figure 1.2). In fact, with this technique, one could design FSS to completely warp the wave around an object and hence make it completely invisible at a specific frequency [8, 9].

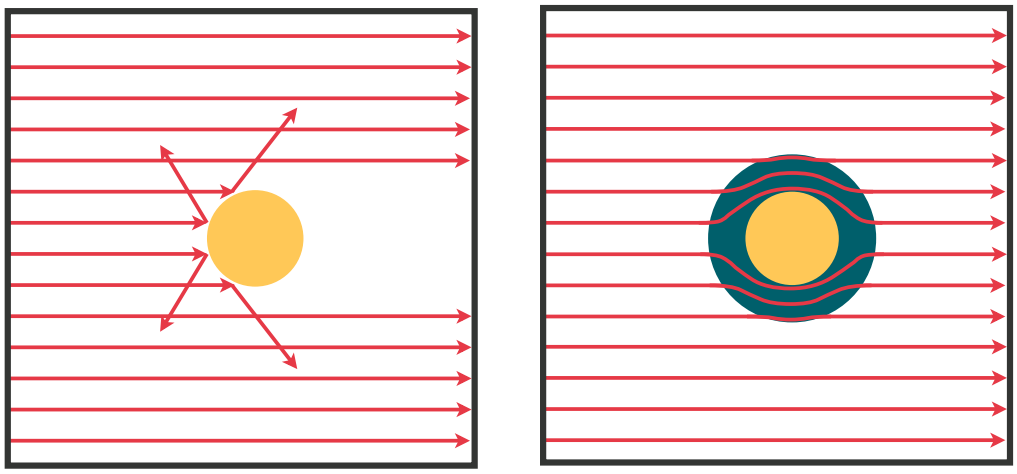


Figure 1.2: Schematic representation of an object (golden circle) without an FSS enclosure (left) and with an FSS enclosure (right). The FSS (teal color) is designed to cloak the object by altering the field distribution. In the left panel, the object scatters the incident wave, whereas in the right panel, the FSS manipulates the wavefronts to guide the electromagnetic waves around the object.

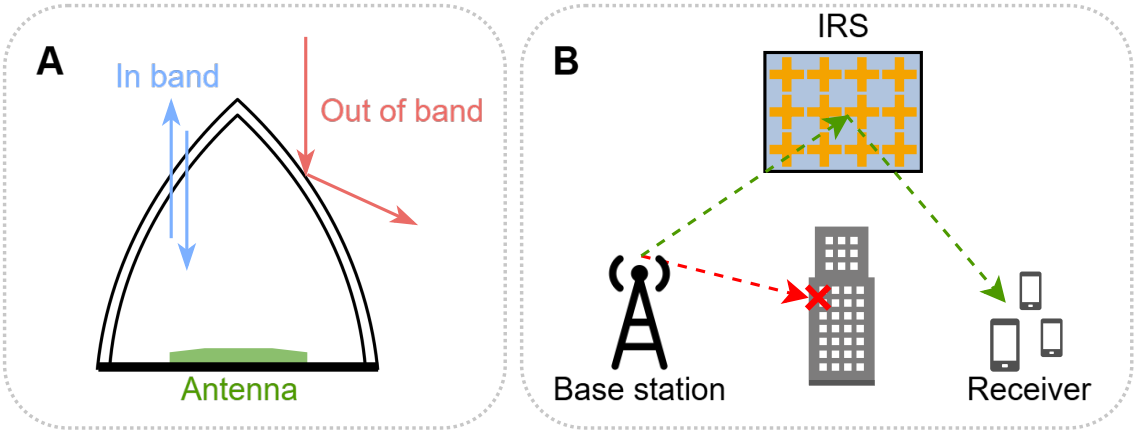


Figure 1.3: Two common applications of FSS. **A**: A stealth radome used in military vehicles. **B**: An intelligent reflecting surface (IRS) that directs the signal from the base station to the receiver, bypassing obstructions.

FSSs are also used in electromagnetic interference shielding. They work as enclosures for electronic systems, such as medical devices or other sensitive equipment, to block interference at specific frequencies. Studies have also shown that it is physically possible to fabricate FSS on flexible substrates, which makes them practical for portable devices and even for integration into automotive interiors [10].

The evolution of wireless communication and advancements in 5G and 6G have increased the need for advanced EM filtering methods. In modern wireless communication systems, intelligent reflecting surfaces (IRS) are employed. These surfaces are composed of FSS elements that control the propagation of EM waves. They can reflect, transmit, and steer (beamform) waves in real time in a controlled manner. This is done to improve coverage, enhance link reliability, and increase data rates [11] (see Figure 1.3B).

Another application is in smart sensing for structural health and environmental monitoring. In this approach, FSSs are fabricated on flexible or textile substrates and integrated into structures or wearable devices. When a structure undergoes strain or deformation, the dimensions of the FSS elements change slightly, which in turn shifts the resonance frequency of the FSS. By monitoring these changes, one can detect mechanical stress or environmental changes in real time. This method is low-profile, non-invasive, cost-effective, and suited for integration into the internet of things [2].

1.4 Related Work

There have been many research articles using machine learning for FSS design. However, most of them consist of researchers coming up with a pattern and then using ML algorithms to optimize the pattern dimensions, that is, the geometric parameters of that specific pattern. While this approach reduces the time required for designing an FSS, it still does not fully harness the power of generative ML to

create novel designs given a desired frequency response.

For example, in the study [12], the authors proposed an initial predefined pattern that was sunflower-inspired. They then trained an ML model (a decision tree algorithm) using various geometric parameters of the pattern and their corresponding frequency responses. Once the model was trained, it was used to predict the optimal geometric parameters given a desired frequency response. A similar study, focusing on a different frequency band and application, used another predefined pattern, a plus-shaped design with a circle in the center [13]. In that work, a simple fully connected neural network (FCNN) was trained to learn the mapping between the geometric parameters of the predefined pattern and the frequency response, and later used to predict the optimal geometric parameters given a desired behavior. There have been many similar studies (for example, [14, 15, 16]) that follow this approach; what differentiates them is mainly the initial predefined pattern, frequency band, and specific application of the FSS.

These approaches, which start with a predefined pattern and then train an ML model to learn the mapping between the geometric parameters and the EM response, streamline the design process by reducing the number of simulation iterations through the interpolating power of the neural network. However, this approach is quasi-automated and still relies heavily on the designer's experience and creativity, for example, in designing the initial pattern, which is commonly quite complex.

In recent years, there have been a few research articles addressing the fully automated inverse design of FSS through generative ML. For example, a study published in 2023 [17] developed an FCNN to predict the pattern (a binary image encoding metal and dielectric substrate) given the desired S-parameters. However, as mentioned, the generative component consisted mainly of an FCNN, which is not inherently a generative model and therefore did not harness the power of recent generative ML algorithms such as generative adversarial networks (GANs), variational autoencoders (VAEs), or diffusion models. Furthermore, that study used binary images of size 30×30 pixels and only 3000 samples of patterns with their corresponding simulated S-parameters in the training data. In a similar study that used 52×52-pixel binary images to represent the pattern, a VAE was used as a generator [18].

While this thesis shares similarities with the aforementioned studies, it differs by utilizing a larger search space and dataset, as well as by adopting one of the more recent generative machine learning architectures: a conditional variational autoencoder (cVAE). Standard VAEs are known to sometimes produce blurry or lower-quality images because of their inherent design. In contrast, the conditional variant (cVAE) leverages auxiliary information, in this case the desired S-parameters, to guide the generation process more precisely. This not only leads to improved image quality and increased fidelity with respect to the target characteristics but also ensures a more stable training process. Moreover, while GANs can be unstable and require constant fine-tuning to avoid issues such as mode collapse, the cVAE framework offers a robust and straightforward alternative without reliance on adversarial training mechanisms.

2

Theory

This chapter presents the mathematical framework for electromagnetic wave propagation in periodic structures. We derive plane-wave solutions from Maxwell's equations, introduce material properties for metals and dielectrics, formulate the Floquet modal expansion for periodic structures, and define the scattering parameters S_{11} and S_{21}. In addition, we present an overview of artificial neural networks and advanced machine learning techniques relevant to our inverse design approach. This includes the architecture and training of feedforward networks, convolutional layers, and a conditional variational autoencoder, which will work together to map the desired electromagnetic responses to optimal FSS patterns.

2.1 Electromagnetic Plane Waves

In a homogeneous, isotropic, source-free medium, Maxwell's curl equations in phasor form are

$$\nabla \times \mathbf{E}(\mathbf{r}) = -j\,\omega\,\mu\,\mathbf{H}(\mathbf{r}), \tag{2.1}$$

$$\nabla \times \mathbf{H}(\mathbf{r}) = j\,\omega\,\varepsilon\,\mathbf{E}(\mathbf{r}), \tag{2.2}$$

where ω is the angular frequency, $\mu = \mu_0\mu_r$ is the permeability, and $\varepsilon = \varepsilon_0\varepsilon_r$ is the permittivity. A plane-wave solution is given by

$$\mathbf{E}(\mathbf{r}) = \Re\Big\{\mathbf{E}_0\, e^{-j\,\mathbf{k}\cdot\mathbf{r}}\Big\}, \quad \mathbf{H}(\mathbf{r}) = \Re\Big\{\mathbf{H}_0\, e^{-j\,\mathbf{k}\cdot\mathbf{r}}\Big\}, \tag{2.3}$$

with the position vector $\mathbf{r} = (x, y, z)$, constant amplitude vectors $\mathbf{E}_0$ and $\mathbf{H}_0$, and wave vector $\mathbf{k}$. The dispersion relation

$$k \equiv |\mathbf{k}| = \omega\,\sqrt{\mu\,\varepsilon} = \frac{\omega}{c}\sqrt{\varepsilon_r\,\mu_r}, \tag{2.4}$$

with $c = 1/\sqrt{\mu_0\varepsilon_0}$, [19, 20].

2.2 Material Models: Metals and Dielectrics

2.2.1 Metals

In FSS applications, metallic elements are often modeled as Perfect Electric Conductors (PECs), satisfying

$$\mathbf{E}_{\parallel}\Big|_{\text{metal}} = \mathbf{0}. \tag{2.5}$$

For real metals with finite conductivity σ (S/m), fields penetrate up to the skin depth

$$\delta = \sqrt{\frac{2}{\omega\,\mu\,\sigma}}, \tag{2.6}$$

where $\mu = \mu_0\mu_r$. If the metal thickness t_{metal} satisfies $t_{\text{metal}} \gg \delta$, the PEC approximation holds [1].

2.2.2 Dielectrics

Dielectric substrates are characterized by a relative permittivity ε_r and a loss tangent $\tan\delta$. Their response is modeled by the complex permittivity

$$\varepsilon = \varepsilon_0\,\varepsilon_r\,(1 - j\,\tan\delta), \tag{2.7}$$

which directly affects the resonance frequency and bandwidth of the FSS [19].

2.3 Periodic Structures and Modal Analysis

An FSS is realized as a two-dimensional periodic array of metallic elements on a dielectric substrate. The lattice translation is defined as

$$\mathbf{R} = m\,p_x\,\hat{\mathbf{x}} + n\,p_y\,\hat{\mathbf{y}}, \quad m, n \in \mathbb{Z}, \tag{2.8}$$

and Bloch's theorem requires that

$$\mathbf{E}(\mathbf{r} + \mathbf{R}) = \mathbf{E}(\mathbf{r})\,e^{\,j\,\mathbf{k}_B\cdot\mathbf{R}}, \tag{2.9}$$

with $\mathbf{k}_B$ being the Bloch wave vector. Consequently, the total field can be expanded into Floquet modes:

$$\mathbf{E}(\mathbf{r}) = \sum_{m,n} \mathbf{E}_{m,n}\,\exp\!\left\{ j\left[\mathbf{k}_0 + \frac{2\pi m}{p_x}\hat{\mathbf{x}} + \frac{2\pi n}{p_y}\hat{\mathbf{y}}\right]\cdot\mathbf{r}\right\}, \tag{2.10}$$

where $\mathbf{k}_0$ is the incident wave vector [21].

2.4 Scattering Parameters

When an incident plane wave with complex amplitude E_{inc} impinges on an FSS unit cell, it produces a reflected wave of amplitude E_{ref} and a transmitted wave

of amplitude E_{trans}, see Figure 2.1 for illustration. The scattering parameters are defined as

$$S_{11} = \frac{E_{\text{ref}}}{E_{\text{inc}}}, \qquad S_{21} = \frac{E_{\text{trans}}}{E_{\text{inc}}}, \tag{2.11}$$

which are complex and thus encode both amplitude and phase information. Their linear magnitudes, $|S_{11}|$ and $|S_{21}|$, represent the amplitude ratios; squaring these values gives the reflected and transmitted power ratios, respectively. An ideal *band-stop* FSS is designed such that $|S_{11}|^2 \approx 1$ and $|S_{21}|^2 \approx 0$ over the stop band, while a *band-pass* FSS requires $|S_{11}|^2 \approx 0$ and $|S_{21}|^2 \approx 1$. In a lossless network, energy conservation demands that

$$|S_{11}|^2 + |S_{21}|^2 = 1. \tag{2.12}$$

While in practice, material losses ensure that $|S_{11}|^2 + |S_{21}|^2 < 1$, [20, 22].

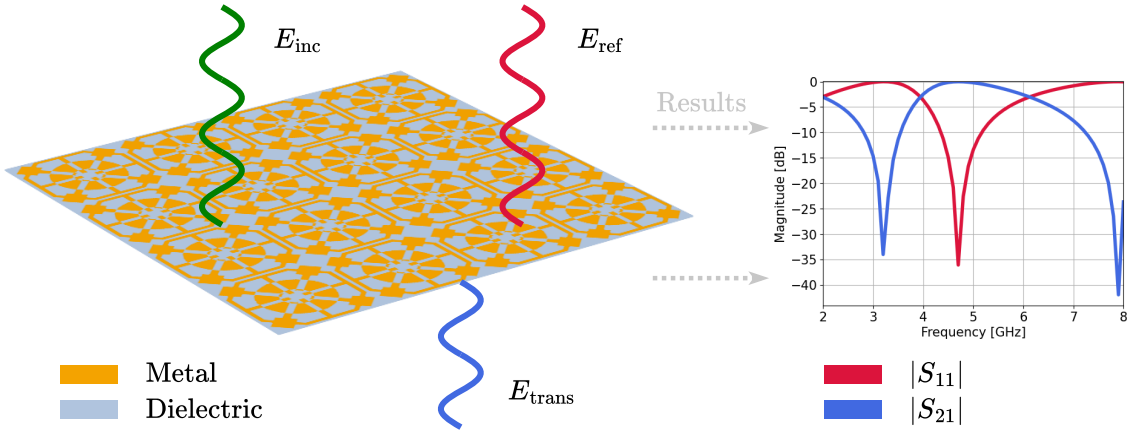


Figure 2.1: A schematic of an FSS array illustrating an incident wave E_{inc} (green), reflected wave E_{ref} (red), and transmitted wave E_{trans} (blue). The metallic pattern (gold) is repeated on a dielectric substrate (light blue). On the right, the simulated scattering parameters' magnitudes are shown as $|S_{11}|$ (red) and $|S_{21}|$ (blue) in dB versus a frequency range.

2.5 Artificial Neural Networks Fundamentals

Artificial neurons, in the context of artificial neural networks (ANNs), are computer-simulated biological neurons. The explanation of ANNs in this chapter is targeted at the classical feedforward neural network (and not other types, such as in reservoir computing). The purpose of ANNs is to be taught, usually through large amounts of data, to accomplish a certain task, typically one that is not easily programmable in an explicit manner, by learning a mapping from inputs to outputs. Each neuron has a *state* (x), a value derived from the neurons in the previous layer through weighted connections, much like how biological neurons communicate via electrical and chemical signals [23]. The feedforward information propagation mechanism in an ANN is defined as follows. Let $\boldsymbol{x}^{(0)}$ be the input layer and $\boldsymbol{O}$ be the output, see Figure 2.2. Denote the number of hidden layers (i.e., the layers between the input and output layers) by N. Then the output is computed by

$$\boldsymbol{O} = \boldsymbol{x}^{(N+1)} = g^{(N+1)}\Big(\boldsymbol{W}^{(N+1)}\boldsymbol{x}^{(N)} - \boldsymbol{\theta}^{(N+1)}\Big), \tag{2.13}$$

and state recursion for the hidden layers is given by:

$$\boldsymbol{x}^{(\ell)} = g^{(\ell)}\Big(\boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell-1)} - \boldsymbol{\theta}^{(\ell)}\Big), \quad \ell = 1, 2, \ldots, N. \tag{2.14}$$

Here, $\boldsymbol{W}^{(\ell)}$ represents the weights connecting layer $\ell-1$ to layer ℓ and $\boldsymbol{\theta}^{(\ell)}$ represents the biases (or thresholds). Note that the linear operations within the parentheses would only allow the network to model linear relationships; hence, a nonlinear *activation function* g is applied to introduce nonlinearity [24]. Some commonly used activation functions include:

$$\begin{aligned}
\textbf{Sigmoid}: &\quad \sigma(z) = \frac{1}{1+e^{-z}},\\
\textbf{Tanh}: &\quad \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}},\\
\textbf{ReLU}: &\quad \mathrm{ReLU}(z) = \max(0,\, z),\\
\textbf{SiLU}: &\quad \mathrm{SiLU}(z) = z\left(\frac{1}{1+e^{-z}}\right) = z\sigma(z),\\
\textbf{Softmax}: &\quad \mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}, \quad i = 1, \ldots, K.
\end{aligned}$$

As mentioned earlier, the network is trained by allowing it to predict an output (via Equations (2.13) and (2.14)), comparing this prediction with the true value through a loss function, and then updating the learnable parameters accordingly

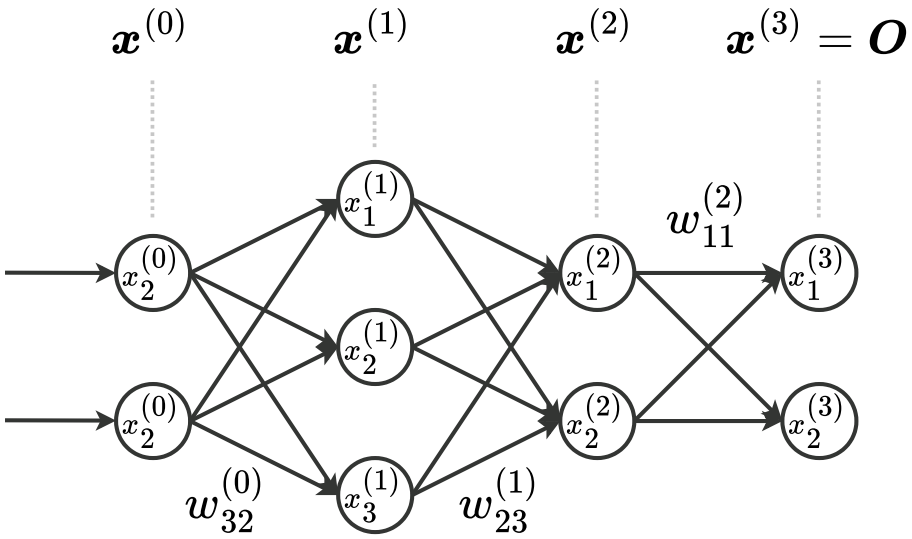


Figure 2.2: A schematic representation of a feed forward neural network with two input neurons, two hidden layers, and two output neurons. The first hidden layer ($\boldsymbol{x}^{(1)}$) consists of three neurons, while the second hidden layer ($\boldsymbol{x}^{(2)}$) has two neurons. The connections between layers are weighted by $w_{j\leftarrow i}^{(\ell)}$, where ℓ indicates the layer index. Example weights, such as $w_{32}^{(0)}$, $w_{23}^{(1)}$, and $w_{11}^{(2)}$, are labeled to illustrate layer connections.

[24]. Common loss functions include:

$$\textbf{Mean Squared Error}: \quad \mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\,,$$

$$\textbf{Binary Cross} - \textbf{Entropy}: \quad \mathcal{L}_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}\Big[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\Big],$$

$$\textbf{Categorical Cross} - \textbf{Entropy}: \quad \mathcal{L}_{\text{CCE}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{i,k}\log \hat{y}_{i,k}.$$

Here, $\hat{y}$ denotes the network's prediction, while y denotes the ground truth value for the i-th sample in a dataset of n samples. The choice of the loss function is problem-specific; for example, MSE may be suitable for a regression task, while cross-entropy functions are appropriate for classification tasks [24].

Given the loss value $\mathcal{L}$, the network's learnable parameters, namely, the weights $\boldsymbol{W}$ and biases $\boldsymbol{\theta}$, are updated according to the gradient descent principle through the algorithm known as *backpropagation*. In essence, backpropagation computes the gradient of the loss function with respect to each parameter by propagating the error backward from the output layer to the input layer.

2.5.1 Backpropagation

To understand backpropagation, we begin by defining the *local field* (or net input) of the neurons. For the output layer (layer $N+1$), the local field is

$$\boldsymbol{B}^{(N+1)} = \boldsymbol{W}^{(N+1)}\boldsymbol{x}^{(N)} - \boldsymbol{\theta}^{(N+1)}.$$

The error at the output is then expressed as

$$\Delta^{(N+1)} = (t - \boldsymbol{O}) \odot g'^{(N+1)}\Big(\boldsymbol{B}^{(N+1)}\Big),$$

where t denotes the target output, $g'^{(N+1)}$ is the derivative of the activation function at the output layer, and $\odot$ indicates the element-wise Hadamard product. Note that since we need to use the derivative of the activation function, such a function must be differentiable [24].

For the hidden layers ($\ell = N, N-1, N-2, \ldots, 1$), the error is computed recursively by applying the chain rule:

$$\Delta^{(\ell)} = \Big(\big(\boldsymbol{W}^{(\ell+1)}\big)^T \Delta^{(\ell+1)}\Big) \odot g'^{(\ell)}\Big(\boldsymbol{B}^{(\ell)}\Big),$$

with

$$\boldsymbol{B}^{(\ell)} = \boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell-1)} - \boldsymbol{\theta}^{(\ell)}.$$

This backward flow of error signals is the origin of the term *backpropagation*.

Once the errors $\Delta^{(\ell)}$ are determined for each layer, the parameters are updated via gradient descent. The weight update for layer ℓ is given by

$$\delta\boldsymbol{W}^{(\ell)} = \eta\, \Delta^{(\ell)} \left(\boldsymbol{x}^{(\ell-1)}\right)^T,$$

and the corresponding bias update is

$$\delta\boldsymbol{\theta}^{(\ell)} = -\eta\, \Delta^{(\ell)},$$

where $\eta > 0$ is the learning rate.

Thus, the updated weights and biases are obtained as [24]:

$$\boldsymbol{W}^{(\ell)} \leftarrow \boldsymbol{W}^{(\ell)} + \delta\boldsymbol{W}^{(\ell)},$$

$$\boldsymbol{\theta}^{(\ell)} \leftarrow \boldsymbol{\theta}^{(\ell)} + \delta\boldsymbol{\theta}^{(\ell)}.$$

For batch training, where the weight and bias updates are averaged over all p training patterns, the updates become

$$\delta\boldsymbol{W}^{(\ell)} = \eta\, \frac{1}{p} \sum_{\mu=1}^{p} \Delta^{(\ell)}(\mu) \left(\boldsymbol{x}^{(\ell-1)}(\mu)\right)^T,$$

$$\delta \boldsymbol{\theta}^{(\ell)} = -\eta \, \frac{1}{p} \sum_{\mu=1}^{p} \Delta^{(\ell)}(\mu).$$

Alternatively, in stochastic gradient descent, the updates are applied after processing each individual pattern:

$$\delta \boldsymbol{W}^{(\ell)} = \eta \, \Delta^{(\ell)}(\mu) \left(\boldsymbol{x}^{(\ell-1)}(\mu) \right)^{T},$$

$$\delta \boldsymbol{\theta}^{(\ell)} = -\eta \, \Delta^{(\ell)}(\mu).$$

It is important to note that the choice of *hyperparameters*, such as the activation function, learning rate, number of neurons, number of hidden layers, etc., is largely an experimental, experience-based fine-tuning decision. The choice of a loss function, the objective, depends heavily on the specific task.

The gradient descent and stochastic gradient descent formulations presented above are typically used in the literature to illustrate the basic process of computing gradients and updating network parameters. In practice, modern machine learning applications generally employ more sophisticated optimizers that incorporate momentum and adaptive learning rates, such as Adaptive Moment Estimation (Adam), a method for stochastic optimization [25].

2.6 Advanced Neural Network Concepts

Now that we have covered the fundamentals in neural networks, feedforward propagation, learning (backpropagation), activation functions, and loss functions, we now introduce several advanced concepts and common layers that are needed for this work. In particular, we focus on convolutional and transposed convolutional layers, as well as residual connections and attention mechanisms.

2.6.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are designed to process data with a grid-like topology, such as images. In a convolutional layer, a convolution (denoted by $\circledast$) is performed between an input image ($\boldsymbol{I}$) and a learnable filter (kernel) ($\boldsymbol{K}$) to produce a feature map [24]. The kernel size is a hyperparameter, and the kernel weights are learnable parameters. Mathematically, the convolution operation is defined as:

$$(\boldsymbol{I} \circledast \boldsymbol{K})\,(i,j) = \sum_{m} \sum_{n} \boldsymbol{I}(i+m, j+n)\, \boldsymbol{K}(m,n).$$

This operation acts as a feature extractor. Transposed convolutional layers reverse this process to upsample feature maps. An illustration of convolution and transposed convolution is provided in Figure 2.3.

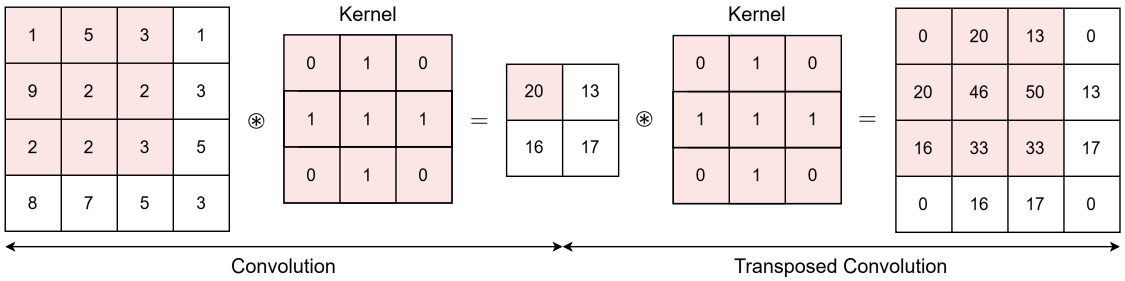


Figure 2.3: Illustration of convolution and transposed convolution. The input map (left-most) is convolved with a kernel to produce a feature map. The feature map is then transposed convolved with a kernel to produce an upsampled feature map (right-most).

2.6.2 Residual Connections

Residual connections help train neural networks by allowing gradients to flow more directly [24]. In a residual block, the input is added to the output of a series of layers. This can be written as:

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}) + \boldsymbol{x},$$

where $\mathcal{F}(\boldsymbol{x})$ represents the layers in the network (typically convolution, activation functions, etc.) applied to $\boldsymbol{x}$. This simple mechanism helps to reduce the vanishing gradient problem [24]. A schematic of a residual block is shown in Figure 2.4.

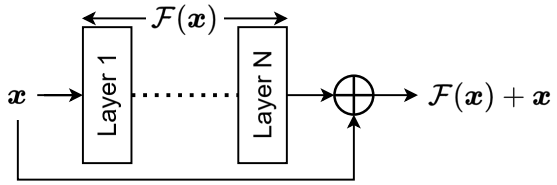


Figure 2.4: Schematic representation of a residual block. The input is added via a skip connection to the output of a series of layers.

2.6.3 Attention Mechanisms

Attention mechanisms allow neural networks to focus on the most relevant parts of the input data. In the scaled dot-product attention, given queries Q, keys K, and values V, the attention operation is defined as [26]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where d_k is the dimension of the key vectors.

Multi-head attention extends this idea by computing multiple attention operations in parallel. This can be written as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O,$$

with each head defined by:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V).$$

In cross attention, the queries come from one sequence while the keys and values come from another in order to allow the model to relate information across different inputs.

2.7 Conditional Variational Autoencoder

A variational autoencoder (VAE) is a generative neural network that learns a probabilistic latent representation of the input pattern P [27]. In this context, the encoder is a distribution $q_\phi(z \mid P)$ that maps each pattern P to a latent variable z, and the decoder is a conditional distribution $p_\theta(P \mid z)$ that attempts to reconstruct P given z. Typically, a prior $p(z)$ (often chosen to be a standard multivariate Gaussian) is imposed on the latent space $\mathcal{Z}$ to regularize the learned representations.

To train such a model, one minimizes the negative of the evidence lower bound (ELBO). Denoting this generator loss by $\mathcal{L}_G$, the objective can be written as:

$$\mathcal{L}_G(\theta, \phi; P) \;=\; -\,\mathbb{E}_{q_\phi(z|P)}\Big[\log\, p_\theta(P \mid z)\Big] \;+\; D_{\text{KL}}\Big(q_\phi(z \mid P) \,\|\, p(z)\Big), \tag{2.15}$$

where θ and ϕ denote the learnable parameters of the decoder and encoder neural networks, respectively, and $D_{\text{KL}}\big(\cdot\|\cdot\big)$ is the Kullback-Leibler divergence, a measure of discrepancy between two probability distributions. The first term in (2.15) penalizes reconstruction errors, while the second term ensures that the approximate posterior $q_\phi(z \mid P)$ remains close to the prior $p(z)$.

A conditional variational autoencoder (cVAE) extends this idea by incorporating additional information $\mathbf{S}$ (for example, class labels or other attributes) into both the encoder and decoder [28]. In our case, $\mathbf{S}$ will represent the simulated S-parameters that characterize the electromagnetic behavior of P. Accordingly, the encoder and decoder become $q_\phi(z \mid P, \mathbf{S})$ and $p_\theta(P \mid z, \mathbf{S})$, respectively. The generator loss becomes

$$\mathcal{L}_G(\theta, \phi; P, \mathbf{S}) \;=\; -\,\mathbb{E}_{q_\phi(z|P,\mathbf{S})}\Big[\log\, p_\theta(P \mid z, \mathbf{S})\Big] \;+\; D_{\text{KL}}\Big(q_\phi(z \mid P, \mathbf{S}) \,\|\, p(z \mid \mathbf{S})\Big). \tag{2.16}$$

In many practical scenarios, $p(z \mid \mathbf{S})$ is kept as the same standard Gaussian used for $p(z)$, thus simplifying the prior to be independent of $\mathbf{S}$.

To enable gradient-based optimization despite the sampling of z from the encoder, the *reparameterization trick* is employed. That is, rather than directly sampling

$z \sim q_\phi(z \mid P, \mathbf{S})$, the encoder produces a mean $\mu(P, \mathbf{S})$ and standard deviation $\sigma(P, \mathbf{S})$, and the latent variable is obtained via

$$z = \mu(P, \mathbf{S}) + \sigma(P, \mathbf{S})\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

This reformulation makes the sampling operation differentiable with respect to μ and σ, allowing backpropagation to update the encoder parameters ϕ.

Figure 2.5 schematically illustrates a cVAE architecture, showing how the condition $\mathbf{S}$ is introduced in both the encoder and the decoder. Through this conditioning, the model can learn a structured latent space z that reflects desired attributes encoded in $\mathbf{S}$, thereby enabling the generation of patterns P from the pattern space $\mathcal{P}$ under specified conditions.

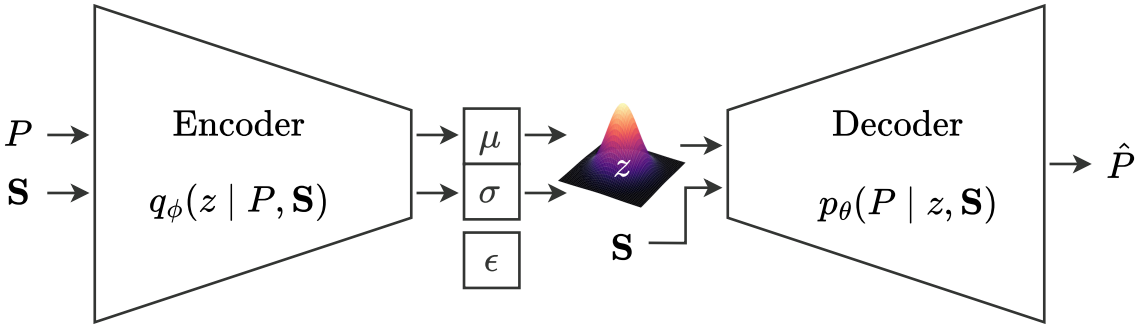


Figure 2.5: Illustration of the cVAE architecture. The encoder maps the pattern P and condition $\mathbf{S}$ to the latent distribution parameters, from which z is sampled via the reparameterization trick. The decoder then reconstructs P conditioned on z and $\mathbf{S}$.

2.8 Theoretical Insights and Constraints

The inverse problem of predicting a pattern $P \in \mathcal{P}$ from desired S-parameters $\mathbf{S} \in \mathcal{S}$ (i.e., the magnitudes $|S_{11}(f)|$ and $|S_{21}(f)|$ over a frequency interval) is inherently nonlinear and non-injective. That is, while the forward mapping $\mathcal{F} : \mathcal{P} \to \mathcal{S}$ assigns a unique $\mathbf{S}$ to each P, its inverse $\mathcal{F}^{-1} : \mathcal{S} \to \mathcal{P}$ is many-to-one; many distinct patterns can yield the same $\mathbf{S}$. Hence, when a generative model produces a candidate pattern $\hat{P}$, it cannot be directly compared to a unique "true" P; instead, the quality of $\hat{P}$ is assessed by evaluating the corresponding (simulated) S-parameters against the desired ones.

3

Methods

In this chapter, we introduce the mathematical notations and general workflow, followed by detailed descriptions of the individual components of the project: data generation, the surrogate model, the generator model, and the optimization process.

3.1 Overall Workflow

Let $\mathbf{S}_t \in \mathcal{S}$ denote the target S-parameter magnitudes (i.e., $|S_{11}(f)|$ or $|S_{21}(f)|$ over a frequency interval). The cVAE (generator) $\mathcal{N}_G : \mathcal{S}, \mathcal{Z} \to \mathcal{P}$ maps the target $\mathbf{S}_t$ and a random noise vector $z \in \mathcal{Z}$ to a candidate pattern:

$$P = \mathcal{N}_G(\mathbf{S}_t, z).$$

The surrogate model $\mathcal{N}_S : \mathcal{P} \to \mathcal{S}$ then predicts $\mathbf{S}$ corresponding to P:

$$\hat{\mathbf{S}} = \mathcal{N}_S(P) = \mathcal{N}_S\Big(\mathcal{N}_G(\mathbf{S}_t, z)\Big).$$

We define a loss function for the optimization process, $\mathcal{L}_{\text{opt}}\Big(\mathbf{S}_t, \hat{\mathbf{S}}\Big)$, which will be specified in detail later. The objective is to obtain an optimal pattern P^* by minimizing $\mathcal{L}_{\text{opt}}$. In practice, since the generator $\mathcal{N}_G$ and surrogate $\mathcal{N}_S$ are fixed at this stage, we achieve this by finding the optimal noisy vector z^* such that

$$z^* = \arg\min_{z \in \mathcal{Z}} \mathcal{L}_{\text{opt}}\Big(\mathbf{S}_t, \mathcal{N}_S(\mathcal{N}_G(\mathbf{S}_t, z))\Big) = \arg\min_{z \in \mathcal{Z}} \mathcal{L}_{\text{opt}}\Big(\mathbf{S}_t, \hat{\mathbf{S}}\Big),$$

and then obtaining the optimal pattern as

$$P^* = \mathcal{N}_G(\mathbf{S}_t, z^*).$$

We update the latent vector z via gradient descent:

$$z_{t+1} = z_t - \eta \nabla_z \mathcal{L}_{\text{opt}}\Big(\mathbf{S}_t, \mathcal{N}_S(\mathcal{N}_G(\mathbf{S}_t, z_t))\Big),$$

with a learning rate $\eta > 0$. The overall latent optimization process is summarized in Figure 3.1.

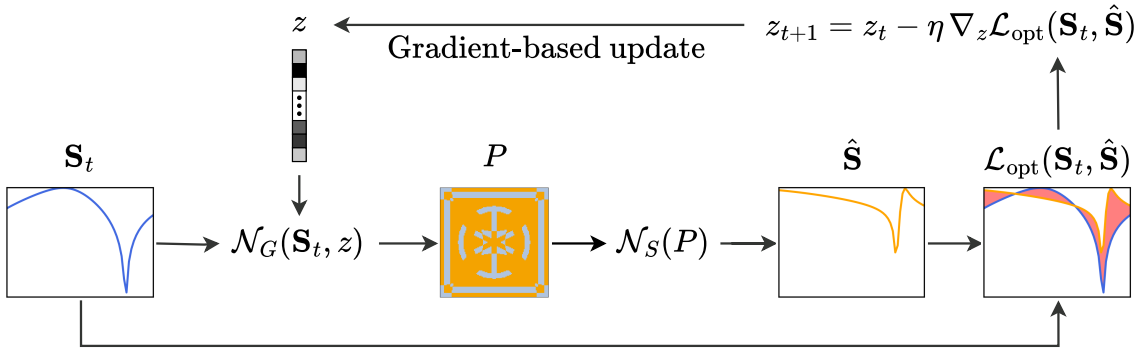


Figure 3.1: Visual schematic of the overall optimization workflow. The diagram shows how the generator $\mathcal{N}_G$ and surrogate $\mathcal{N}_S$ are connected within the optimization loop to produce the optimal pattern P^*.

Throughout the project, only the reflection will be considered. The rationale behind this is that lossless networks are being dealt with, and if a desired behavior is expressed in transmission, it can be converted to reflection through Equation 2.12. This is also the reason why only one curve is displayed in Figure 3.1 as $\mathbf{S}_t$ and $\hat{\mathbf{S}}$.

In the following sections, the following steps will be described: (1) the generation of the dataset $\{(P, \mathbf{S})\}$, (2) the training of $\mathcal{N}_S$ to predict S-parameters from a pattern, (3) the training of $\mathcal{N}_G$ to generate patterns conditioned on S-parameters, and (4) how these networks are combined in an optimization loop to produce the optimal pattern P^*.

3.2 Dataset Creation

In order to train our machine learning models later, we need to generate a dataset of reasonable FSS unit cell patterns and their corresponding frequency responses, $(P, \mathbf{S})$. Due to time limitations, we could only generate and simulate 10,000 patterns.

3.2.1 Pattern Generation

To represent a pattern in an image, we encode metal as 1 and dielectric as 0, so P is a binary image. To balance computational resources, efficiency, and practicality, the image size is chosen to be 128×128 pixels. However, the search space is only binary 64×64 because only one quadrant of the image needs to be decided, and the full pattern is created through mirroring around the x and y axes (see Figure 3.2). In this approach, only the first quadrant of 64×64 pixels is determined, and the full 128×128 pixels pattern is generated through double mirroring. The complete 128×128 pixels pattern is what is simulated later to obtain the S-parameters, while only the first quadrant is used in the machine learning models, since one quadrant uniquely maps to the full pattern. This reduces the image size by a factor of four. Therefore, when the generative ML model predicts a pattern, it essentially predicts the first quadrant, which is then mirrored manually to obtain the complete unit cell

pattern. The rationale behind this approach is to create a relatively small search space for the ML models (64×64 pixels, corresponding to a search space of $2^{64\times64}$), while using a larger image for simulation to achieve a higher resolution and a less rasterized image, enabling smoother modeling of round shapes.

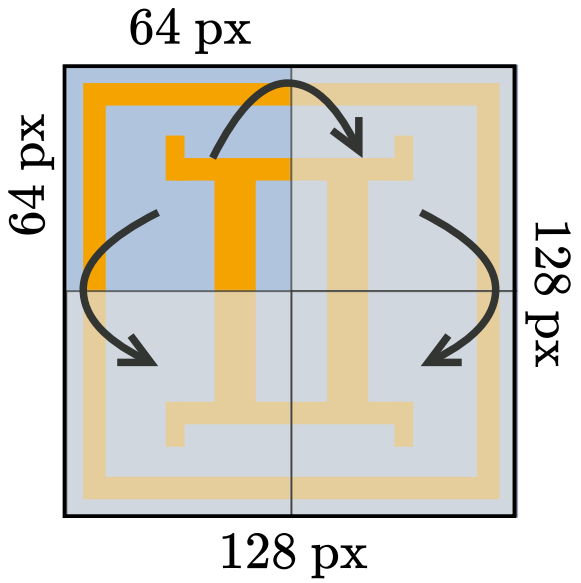


Figure 3.2: Illustration of full unit cell pattern creation through double mirroring.

Generating 10,000 unique patterns manually is not feasible. To address this, we drew 100 unique, realistic, research-inspired patterns and then applied various algorithms to mutate them into 10,000 patterns. These 100 initial patterns were inspired by several research articles and books to mimic common designs used in FSS. An arbitrary sample of these patterns is shown in Figure 3.3.

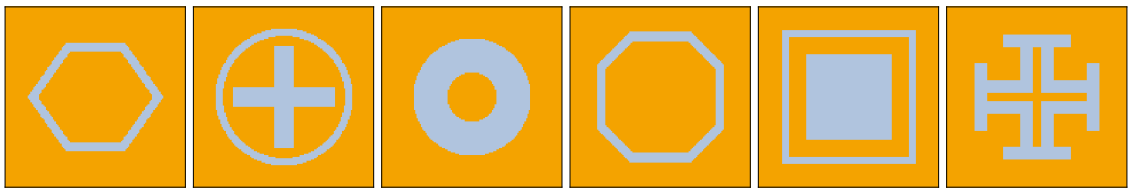


Figure 3.3: Some of the handmade initial patterns.

With the initial 100 patterns in hand, we developed the following algorithms to expand the dataset: operations on single patterns (`inversion`, `rotation 90deg` and morphological operations such as `dilation`, `erosion`, `closing`, `opening`) and operations on pairs of patterns (`OR`, `AND`, `XOR`). The idea behind the `invert` and `rotation 90deg` operations is that if a pattern is realistic, its inverse and rotated version are likely to be realistic as well. Morphological operations were used to teach the ML model how small changes in patterns (for example, a cross with different thicknesses) affect the S-parameters. The binary operations helped to

increase diversity by creating entirely new patterns. Visualization of the outcomes of these different algorithms is shown in Figure 3.4.

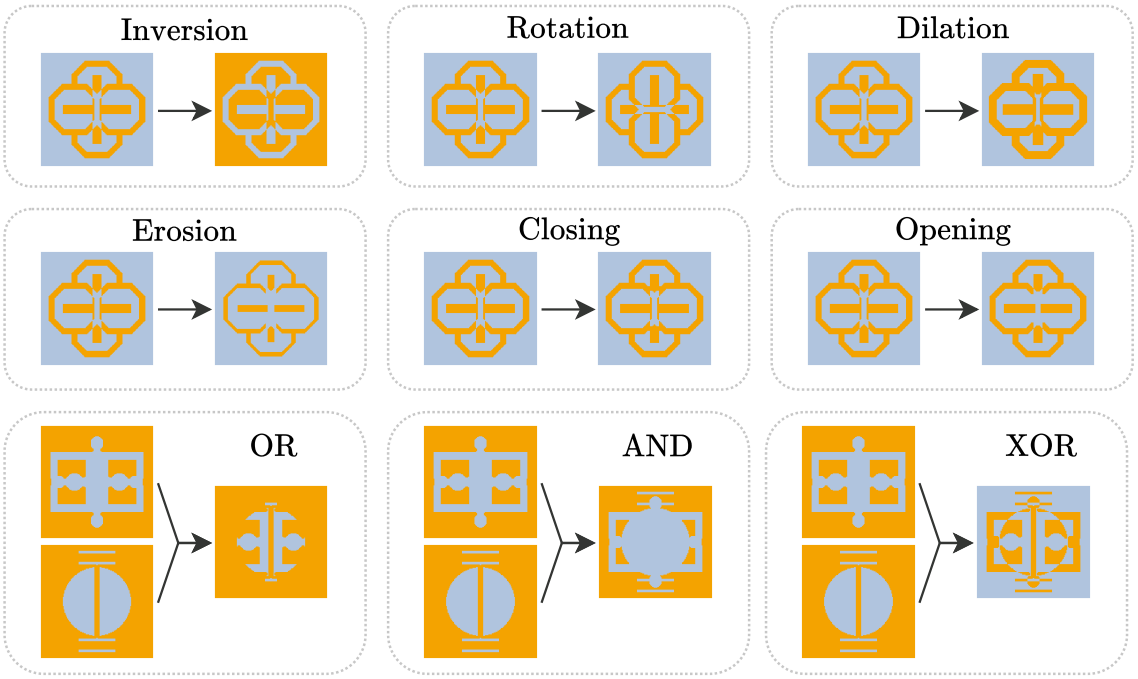


Figure 3.4: Different algorithms used to expand the initial 100 patterns to 10,000.

Inversion followed by rotation was applied to all initial patterns. Then, morphological operations were applied to randomly selected images, and binary operations were applied to randomly selected pairs of images. Additionally, to maximize diversity, a random sequence of operations (e.g., dilation followed by opening applied to an already XORed image) was performed. These random mutations continued until we had 10,000 unique patterns, with duplicates removed.

Now we have 10,000 binary images representing different unit cell patterns.

3.2.2 Simulation

With the complete set of patterns ready, we simulated each one to obtain the corresponding S-parameters. For this, we used Ansys Electronics 2024 R2.1, a high-frequency structure simulator (HFSS), over the frequency interval $[2, 8]$ GHz with steps of 0.1 GHz, which covers widely used bands (S-, and C-band [29]). First, we created a fixed setup - a vacuum network with a dielectric plate in the middle (see Figure 3.5A). The ports were set as Floquet to idealize the infinite duplication of a unit cell pattern, and excitations were applied to the walls. The dielectric used was Rogers RT/duroid 5880, a common industry material with a low dielectric constant ($\varepsilon_r = 2.2$) and a low dissipation factor ($\tan\delta = 0.0009$). This is the plate on which the metal (PEC) is printed. The dielectric thickness was set to 1.575 mm, a standard value [30], and the unit cell dimensions were 15×15 mm. This size was chosen based on the thumb rule of having a side length smaller than half the wavelength at

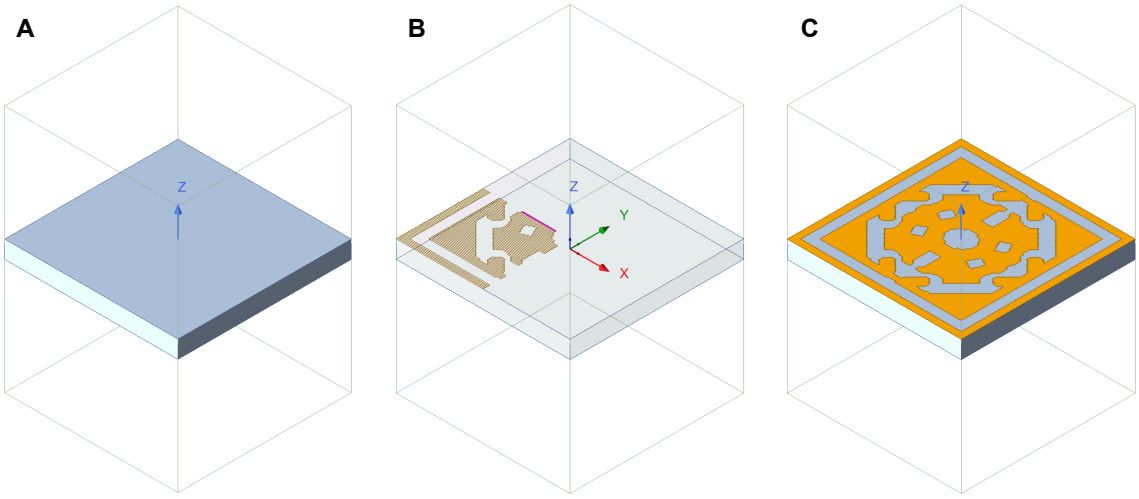


Figure 3.5: Overview of Ansys HFSS setup. **A**: A vacuum network with only a dielectric plate in the middle. **B**: A quadrant of a pattern loaded from a file. **C**: The quadrant in B has been mirror-duplicated around the y and x axes to form the complete pattern.

the highest frequency [31] (for 8 GHz, $\lambda/2 = c/2f \approx 18.75$ mm). Given the plate dimensions of 15×15 mm and the image resolution of 128×128 pixels, each pixel corresponds to approximately $15/128 \approx 0.12$ mm.

To automate the process, an automation script was written using Ansys HFSS's automation framework. The script reads a pattern, recreates its first quadrant in the program (see Figure 3.5B), duplicates and mirrors it around the y and x axes (see Figure 3.5C), runs the simulation, saves the $|S_{11}(f)|$ and $|S_{21}(f)|$ (in steps of 0.1 GHz) data to a CSV file, clears the scene, and repeats the process for all patterns.

The simulation time varied depending on the complexity of each pattern, averaging about 4 minutes per pattern. Running simulations on two machines in parallel, the entire process took roughly 2 weeks of simulation time. As a result, we now have a dataset consisting of 10,000 $(P, \mathbf{S})$ samples.

3.3 Surrogate Model

This is the surrogate model that will predict S-parameters given a pattern P, which will be noted as $\mathcal{N}_S(P) = \hat{\mathbf{S}}$. We will train this model using the complete dataset of 10,000 samples, with splits according to train/validation/test being 0.8/0.1/0.1. The patterns are binary, hence no normalization is needed; the S-parameters, however, will be normalized. The train and validation data will use the same Huber loss

$$\mathcal{L}_S = \begin{cases} 0.5\,(S - \hat{S})^2/\delta, & \text{if } |S - \hat{S}| < \delta, \\ |S - \hat{S}| \;-\; 0.5\,\delta, & \text{otherwise,} \end{cases} \tag{3.1}$$

where $\delta = 0.2$. The rationale behind the choice of δ and the loss function is to penalize larger errors (using an L1 term) since peaks in S-parameters are of most

importance to predict correctly.

The test data will be evaluated with MAE loss $\left|S_i - \hat{S}_i\right|$, as it is more interpretable. For the test data, several other benchmarks will be performed, such as the distribution of count versus absolute error and absolute error versus frequency. The validation data will be used both to save the model with the lowest validation loss over the training epochs and to perform a fine-tuning of the learning rate, batch size, weight decay, and architectural hyperparameters (dropout rate and number of filters in the convolutional residual blocks) over the following search space:

- Learning Rates: $\{0.00005, 0.0005, 0.0001\}$
- Weight Decays: $\{0.00005, 0.0005, 0.0001\}$
- Batch Sizes: $\{16, 32, 64\}$
- Dropout Rates: $\{0, 0.2, 0.4, 0.6\}$
- Filter Configurations: with baseline filters (32, 64, 128), the search space is defined as $\{(32\alpha,\ 64\alpha,\ 128\alpha) \mid \alpha \in \{0.5,\ 1.0,\ 2.0\}\}$

Since the total number of hyperparameter combinations is 324, only 150 randomly selected combinations will be used for fine-tuning, which is done for 50 epochs per hyperparameter configuration, as training with all combinations would require a very long time.

The optimizer Adam will be used. The total number of epochs for the final training using the optimal hyperparameters is 200; however, as mentioned earlier, the best model during these epochs will be saved as the final model.

The architecture for this model is a straightforward convolutional-MLP. The input is a single-channel (binary pattern) 64×64 px image, hence the convolutional layers at the beginning, and the prediction consists of the S-values, hence the MLP at the end. The architecture also includes residual connections to mitigate the risk of vanishing gradients. The architecture is illustrated in Figure 3.6.

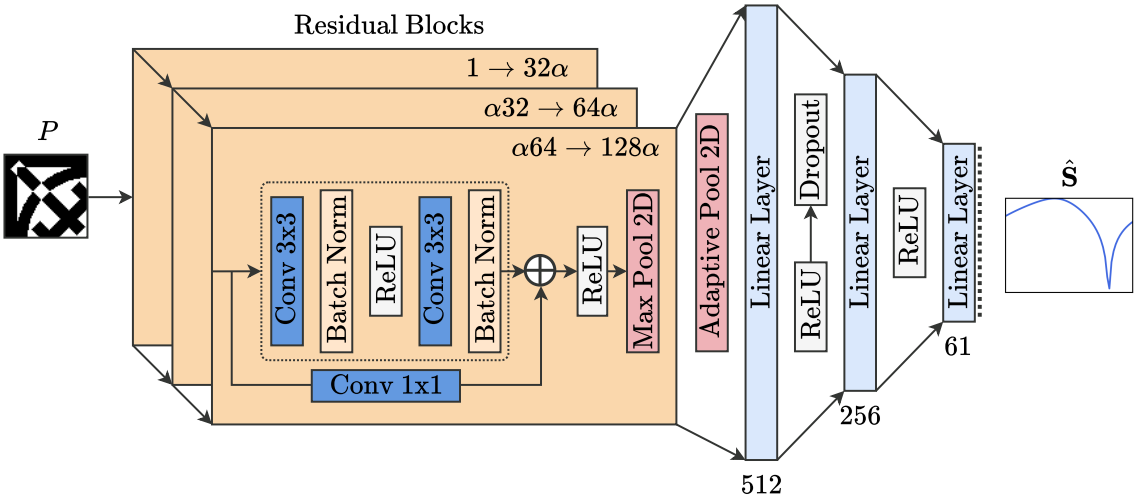


Figure 3.6: Neural network architecture of the surrogate model $\mathcal{N}_S$, with exemplified input P and output $\hat{\mathbf{S}}$.

3.3.1 Dataset Size Analysis

We have a total of 10,000 samples. Out of these, 2,000 samples are reserved for validation and remain constant throughout all experiments. The remaining 8,000 samples are used for training, where we experiment with different training set sizes: 1,000, 2,000, 3,000, ..., and 8,000 samples.

For each training set size, the model is trained for 100 epochs using fixed hyperparameters. The performance proportional measure, denoted as $\mathcal{L}_{S,\text{val}}^{(\text{min})}$, is defined as the minimum validation loss observed over the 100 epochs.

To ensure the robustness of the results, the entire experiment is repeated 5 times with different random selections of the training samples. The final reported performance for each training set size is obtained by averaging the results from these 5 runs. This analysis provides insight into the model's performance as the size of the training dataset increases and helps determine whether our dataset of 10,000 samples is sufficient.

3.4 Generator Model: cVAE

The goal of the generator model is to generate a pattern P, given the condition of the target scattering parameters $\mathbf{S}_t$ and a random sample from the latent space z. The ideal case is that the generated pattern actually yields the conditioned frequency response when simulated. However, to truly ensure that it does so, it will later be fed into the surrogate model for evaluation; this process is described in the next section. The generator model, $\mathcal{N}_G(\mathbf{S}_t, z) = P$, will work as a component in the optimization loop defined later, where its main task is to provide P given $\mathbf{S}_t$ and the latent vector, which is the parameter that will be optimized in order to yield the optimal pattern P^*.

This model will be trained using the complete dataset of 10,000 samples with splits according to train/validation being 0.8/0.2. Similarly to the data preprocessing done in the surrogate model, the patterns will not be normalized as they are already binary, and the S-parameters will be normalized. The train and validation data will use the same VAE loss

$$\mathcal{L}_G = \underbrace{-\sum_{i=1}^{n}\Big[P_i \log \hat{P}_i + \big(1-P_i\big)\,\log\big(1-\hat{P}_i\big)\Big]}_{\text{Binary Cross-Entropy}} \underbrace{-\tfrac{1}{2}\sum_{j=1}^{D_L}\Big[1+\log\,\sigma_j^2\;-\;\mu_j^2\;-\;\sigma_j^2\Big]}_{\text{KL-Divergence}}$$

Here, P_i denotes the i-th pixel of the ground-truth pattern $P \in \{0,1\}^n$, $\hat{P}_i$ is the predicted probability of that pixel being 1, and μ_j, σ_j are the parameters of the latent distribution for the j-th dimension of the latent space which has a total dimension of D_L (a hyperparameter).

The validation data will be used both to save the model with the lowest validation loss over the training epochs and to perform a fine-tuning of the learning rate, batch size, latent dimension, and number of heads in multi head self attention over the following search space:

- Learning Rates: $\{0.00001, 0.0001, 0.0005\}$
- Batch Sizes: $\{32, 64\}$
- Latent Dimensions: $\{256, 512, 1024\}$
- Number of Heads: $\{2, 4, 6\}$

The total number of hyperparameter combinations is 54 and all combinations will be evaluated for fine-tuning, which is done for 70 epochs per hyperparameter configuration. The optimizer Adam will be used. The total number of epochs for the final training using the optimal hyperparameters is 200.

The architecture for this model is a conditional variational autoencoder. The encoder part consists of four convolutional layers of kernel size 4×4, each followed by the ReLU activation function. The encoder also includes a multi head attention (MHA) module at the end. The decoder is the reverse of the encoder, consisting of transposed convolutional layers, except that it does not contain an MHA, and the final activation function is a Sigmoid to bound the pixel value prediction to $[0, 1]$. Directly after the prediction is done, a binarization is applied to the image manually; this is not illustrated in the figure below and is only used during inference (not during training). The architecture is illustrated in Figure 3.7. When the cVAE is trained, only the decoder will be extracted and used in the optimization loop in the next step.

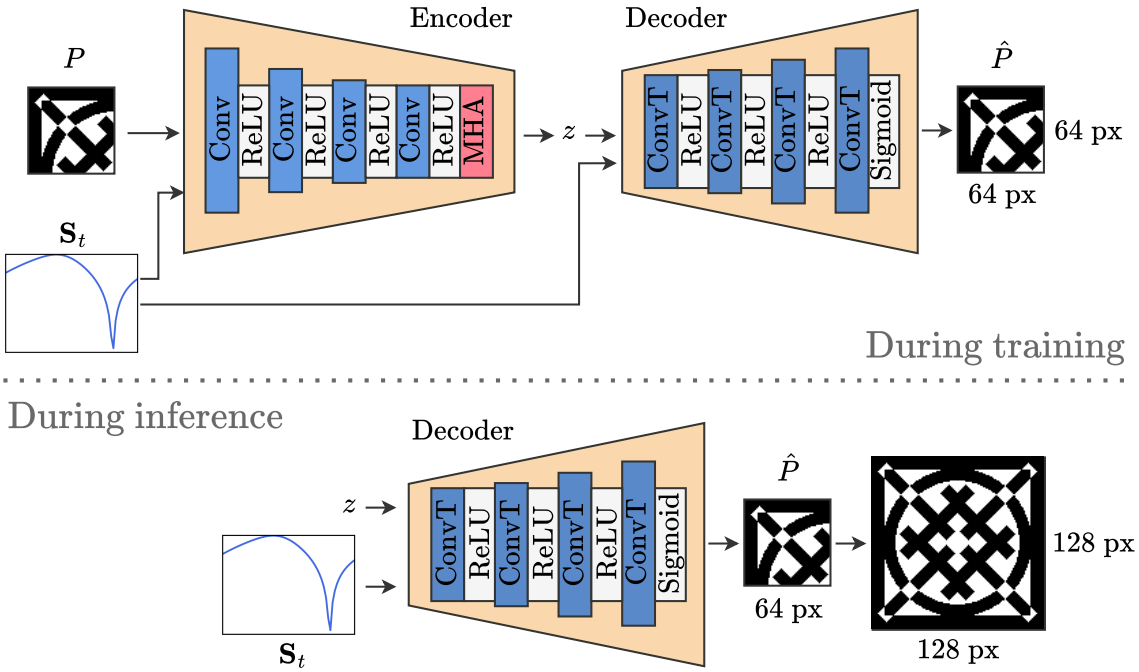


Figure 3.7: Neural network architecture of the generator model $\mathcal{N}_G$ (cVAE), with exemplified input $\mathbf{S}_t$ and output $\hat{P}$.

3.5 Optimization

The optimization process is the final step in our workflow, aiming to generate an optimal pattern P^* that produces a frequency response closely matching the desired target $\mathbf{S}_t$. The workflow is illustrated in Figure 3.1. To achieve this, we iteratively adjust the latent variable z used by the generator model $\mathcal{N}_G$ until the surrogate model $\mathcal{N}_S$ predicts an S-parameter response $\hat{\mathbf{S}}$ that is close enough to $\mathbf{S}_t$.

Our objective is formulated as minimizing the loss function $\mathcal{L}_{\text{opt}}(\mathbf{S}_t, \hat{\mathbf{S}})$, which is defined as the MSE between $\mathbf{S}_t$ and $\hat{\mathbf{S}}$ plus a ReLU penalty term applied at the frequency index corresponding to the valley (resonant frequency) of the target response (denoted v):

$$\mathcal{L}_{\text{opt}}(\mathbf{S}_t, \hat{\mathbf{S}}) \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big(S_{t,i} - \hat{S}_i\Big)^2 + \max\Big(0,\, \hat{S}_v - S_{t,v}\Big),$$

where N is the number of frequency points. The loss function $\mathcal{L}_{\text{opt}}$ is designed to capture both the overall deviation between the target and predicted S-parameters (through the MSE term) and to emphasize the accuracy at the valley frequency (resonant frequency) v (via the ReLU penalty term).

Since both the generator and surrogate models are fixed during the optimization, the process is performed solely over the latent variable z. In other words, we search for

$$z^* = \arg\min_{z \in \mathcal{Z}} \mathcal{L}_{\text{opt}}\Big(\mathbf{S}_t, \mathcal{N}_S\big(\mathcal{N}_G(\mathbf{S}_t, z)\big)\Big),$$

and then obtain the optimal pattern via

$$P^* = \mathcal{N}_G(\mathbf{S}_t, z^*).$$

The optimization is carried out using gradient descent with the Adam optimizer. Starting from an initial random latent vector z_0, we update the latent variable iteratively:

$$z_{t+1} = z_t - \eta \, \nabla_z \mathcal{L}_{\text{opt}}\Big(\mathbf{S}_t, \mathcal{N}_S\big(\mathcal{N}_G(\mathbf{S}_t, z_t)\big)\Big),$$

where $\eta > 0$ is the learning rate. The use of the Adam optimizer ensures robust convergence in the high-dimensional latent space [25].

When the optimization is complete and an optimal pattern P^* is achieved, it will be modeled in Ansys HFSS to simulate it and obtain the true $\mathbf{S}$, which will be compared with the target $\mathbf{S}_t$. This benchmarking will be performed on 10 different $\mathbf{S}_t$. These $\mathbf{S}_t$ are shown as blue curves in Appendix A.1.

4

Results

4.1 Surrogate Model

The hyperparameter search yielded the following optimal hyperparameters: learning rate $= 5 \cdot 10^{-4}$, weight decay $= 5 \cdot 10^{-5}$, batch size $= 32$, dropout $= 0.4$, and filter scale $= 2$. The train and validation loss using the optimal and worst hyperparameter configurations is shown in Figure 4.1, highlighting the need for hyperparameter fine-tuning. The worst hyperparameter configuration was: learning rate $= 10^{-4}$, weight decay $= 5 \cdot 10^{-4}$, batch size $= 16$, dropout $= 0.6$, and filter scale $= 1$.

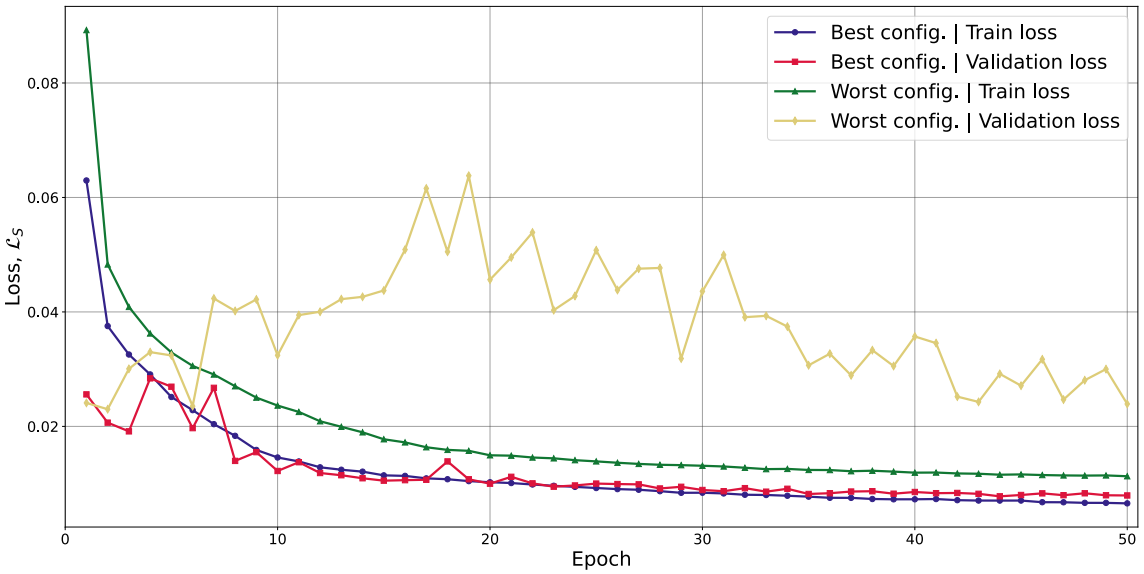


Figure 4.1: Results from hyperparameter fine-tuning of the surrogate model. Train and validation loss over 50 epochs for the best and worst hyperparameter configurations.

Using the optimal hyperparameters, the surrogate model was trained over 200 epochs. However, the model with the lowest validation loss over these epochs was saved as the final model. As seen in Figure 4.2, the final model corresponds to the epoch 63. The train and validation loss are depicted in Figure 4.2, where a steady decrease is observed, which later stagnates.

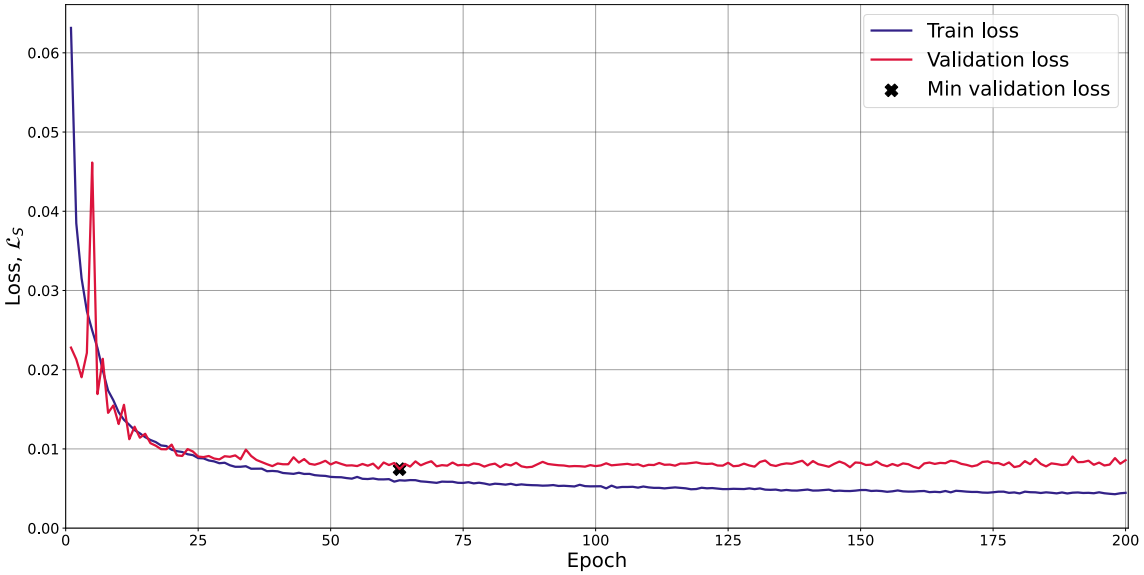


Figure 4.2: Train and validation loss of the surrogate model using the optimal hyperparameters. The marker **x** indicates the minimum of the validation loss curve, i.e., the epoch at which the final model is saved, epoch 63.

4.1.1 Benchmarking

The trained model was evaluated using all 1000 test samples (10% of the total dataset of 10,000). It should be noted that this data was never exposed to the model during training, and is therefore equivalent to completely new patterns. In Figure 4.3, the number of indices versus the absolute error is plotted as a histogram. The count indices represent the number of frequency indices; since every sample covers the frequency range from 2 to 8 GHz, and the data was simulated in steps of 0.1 GHz, there are 61 indices per sample. The distribution exhibits a very steep negative exponential shape, which is desirable as it shows that most indices have low errors. However, it should also be noted that the absolute error axis extends up to 40 dB, which is a very large error; only a few indices exhibit such high errors. In Figure 4.4, the mean absolute error (in dB) over the frequency interval from 2 to 8 GHz is plotted. In this context, the mean refers to the average error for a specific index across all test samples. The curve shows overall low average errors, ranging from 0.5 to 1.9 dB, with a clear trend that higher frequencies (>5 GHz) exhibit higher errors compared to lower frequencies (<5 GHz).

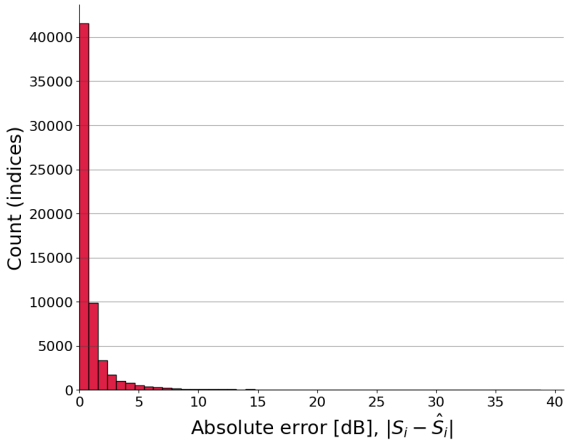


Figure 4.3: Count versus absolute error (in dB) distribution, illustrating that most indices exhibit very low errors, with a sharp decline in count as absolute error increases.

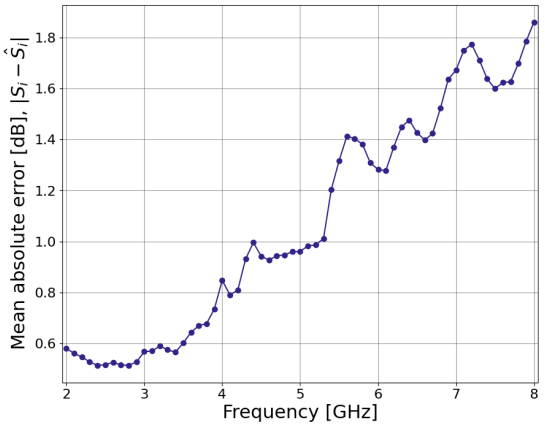


Figure 4.4: Mean absolute error over the test samples across the frequency interval, showing generally low errors with a positive trend of increased errors at higher frequencies.

4.1.2 Show Cases

Figure 4.5 displays three arbitrarily selected samples from the test data, which allow for a visual inspection of the surrogate model predictions alongside the actual ground truth. The samples on the left and in the middle exhibit very different characteristics, yet the predictions are nearly perfect. In the sample on the right, the prediction is accurate at the beginning and end, but the valley is missed.

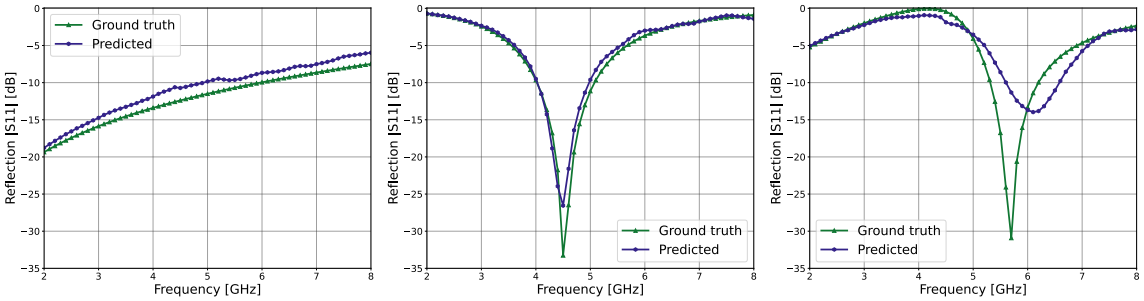


Figure 4.5: Three different samples from the test data showing prediction (blue) and the ground truth (green).

4.1.3 Dataset Size Analysis

The dataset size analysis using the performance metric $\mathcal{L}_{S,\text{val}}^{(\min)}$ yielded the results shown in Figure 4.6.

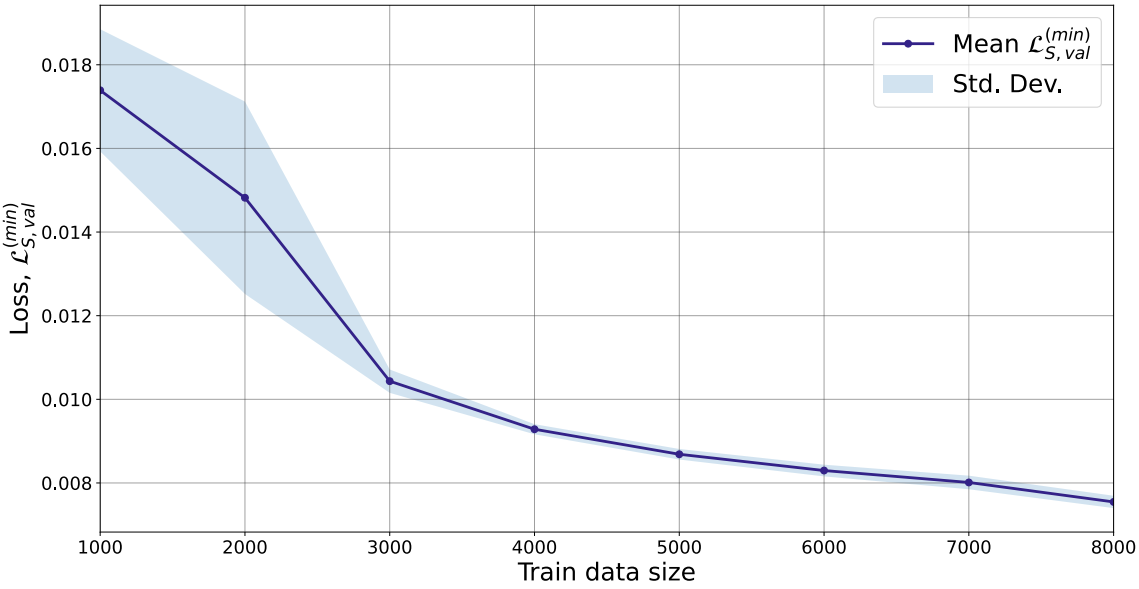


Figure 4.6: Validation loss $\mathcal{L}_{S,\text{val}}^{(\min)}$ on a validation set of 2,000 samples versus training data size ranging from 1,000 to 8,000.

Figure 4.6 clearly indicates that the surrogate model improves with increased training data. The performance metric shows a rapid initial decline that gradually levels off; however, it has not completely plateaued at the highest training data size, suggesting that further increases in the dataset would result in noticeable improvements. The significant reduction in the standard deviation as data increases implies that repeated runs of the data are more likely to yield similar results, which indicates higher reliability of the trained model.

4.2 Generator Model

The hyperparameter search yielded the following optimal hyperparameters: learning rate = 10^{-4}, batch size = 32, latent dimension = 512, and number of heads = 8. The train and validation loss using the optimal and worst hyperparameter configurations is shown in Figure 4.7, highlighting the need for hyperparameter fine-tuning. The worst hyperparameter configuration was: learning rate = 10^{-5}, batch size = 64, latent dimension = 1024, and number of heads = 8.

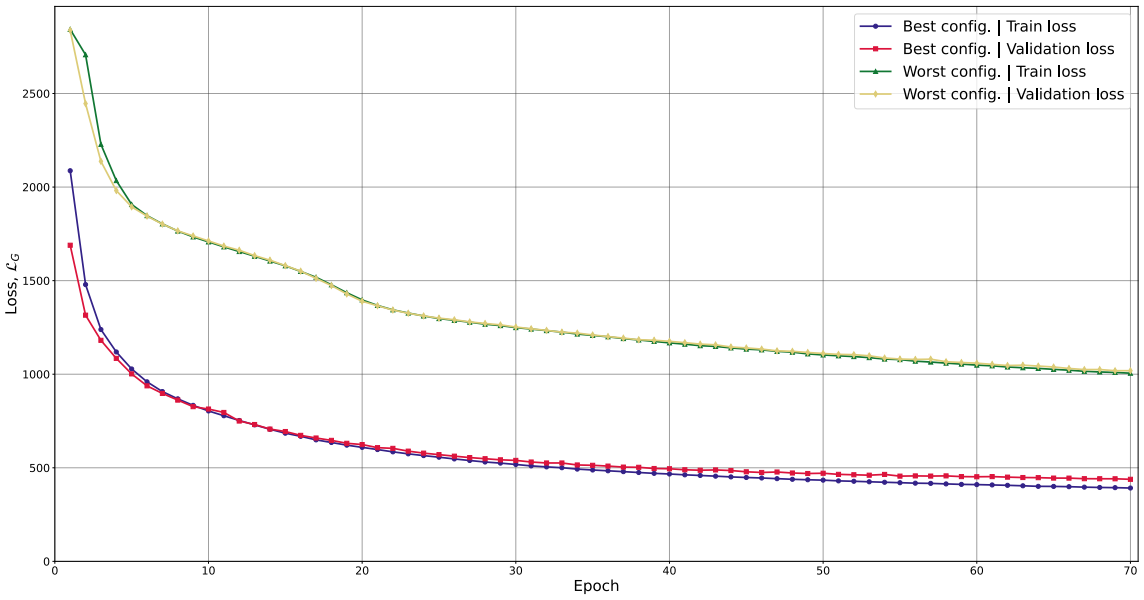


Figure 4.7: Results from hyperparameter fine-tuning of the generator model (cVAE). Train and validation loss over 70 epochs for the best and worst hyperparameter configurations.

Using the optimal hyperparameters, the generator model was trained over 200 epochs. However, the model with the lowest validation loss over these epochs was saved as the final model. As seen in Figure 4.8, the final model corresponds to epoch 107. The train and validation loss are depicted in Figure 4.8, where a steady decrease is observed, which later plateaus.

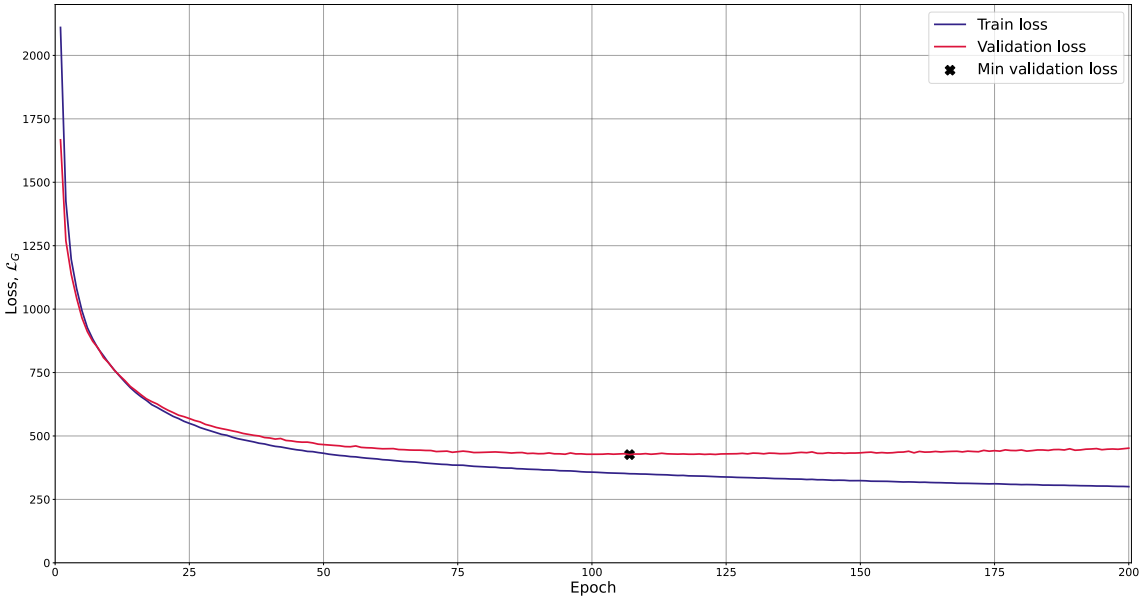


Figure 4.8: Train and validation loss of the generator model using the optimal hyperparameters. The marker **x** indicates the minimum of the validation loss curve, i.e., the epoch at which the final model is saved (epoch 107).

4.2.1 Show Cases

Using the trained cVAE, inference was performed with a random latent vector and arbitrarily chosen conditions from the simulated data. This demonstration serves to show that the model can generate novel, unique patterns beyond those contained in the dataset. A set of outputs is presented in Figure 4.9. It should be noted that this is not a benchmarking nor an optimal design for any desired electromagnetic behavior, but rather a showcase of the outputs.

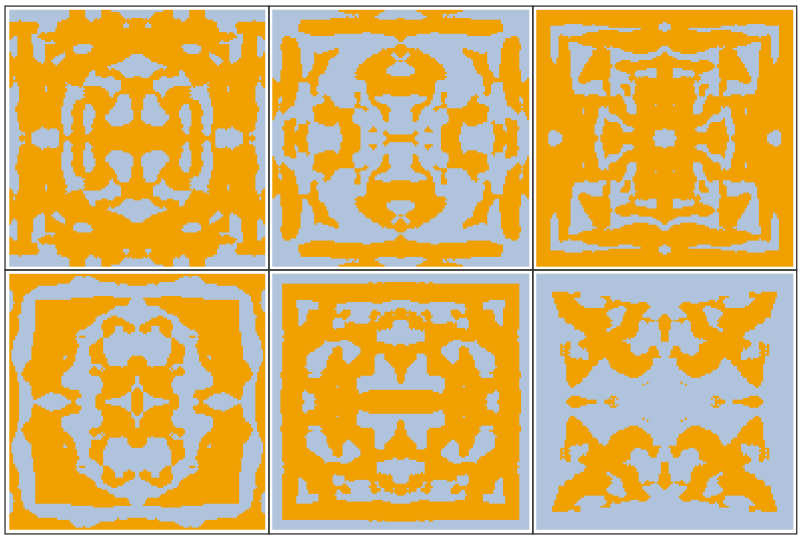


Figure 4.9: A set of 6 arbitrary outputs from the generator model, illustrating its ability to generate novel FSS unit cell patterns.

4.3 Optimization

The optimization was performed using 10 different target S-curves, $\mathbf{S}_t$ (blue curves in Appendix A.1). The results are presented in Appendix A.1. In this appendix, three S-curves are shown: the target S-curve (blue), the predicted S-curve for the optimized pattern (orange), and the true S-curve (green) obtained from simulating the optimized pattern in ANSYS HFSS. Appendix A.2 displays the 10 optimized patterns. Note that the optimization outputs are binary and represent only the top-left quadrant of a complete pattern. In Appendix A.2, the quadrant patterns have been manually colored in gold and blue to visually represent the metal and dielectric surfaces, respectively, and then expanded to illustrate the complete pattern.

From Appendix A.1, we clearly see that our optimization provides good results on samples 1, 3, 5, 6, and 9. On samples 2, 4, 7, 8, and 10, the model performs poorly; however, we still observe a valley in the simulated S-curve, although it is shifted by up to 1.3 GHz from the desired valley.

In Figures 4.10 and 4.11 we show two of the good samples, sample 5 with ($\Delta f = 0.1$ GHz, ΔMag. $= 0.36$ dB) and sample 6 with ($\Delta f = 0.2$ GHz, ΔMag. $= 6.4$ dB).

In Figures 4.12 and 4.13 we show two of the poor samples, sample 7 with (Δf = 0.5 GHz, ΔMag. = 4.6 dB) and sample 8 with (Δf = 0.4 GHz, ΔMag. = 3.0 dB). These values are between $\mathbf{S}_t$ and $\mathbf{S}$ at their respective minima.

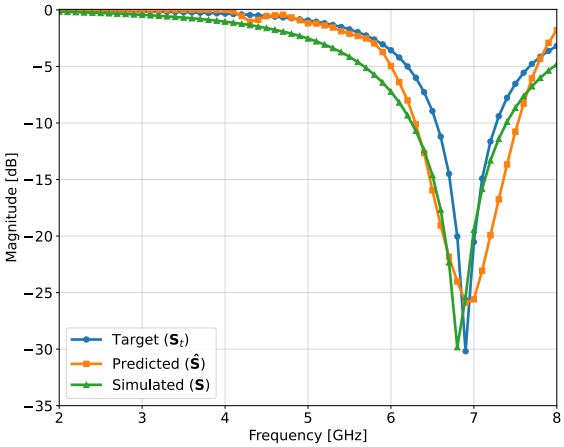


Figure 4.10: Sample 5 - target, predicted, and simulated S-curves

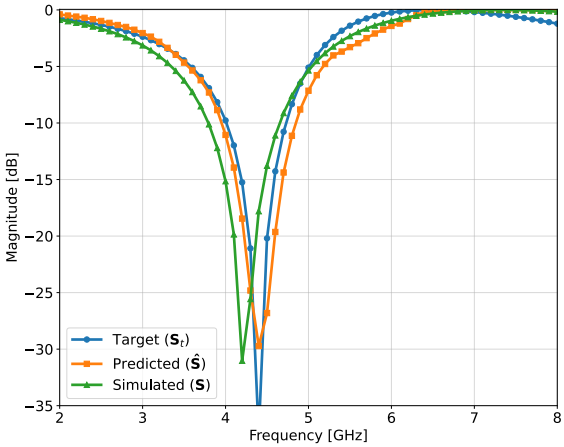


Figure 4.11: Sample 6 - target, predicted, and simulated S-curves

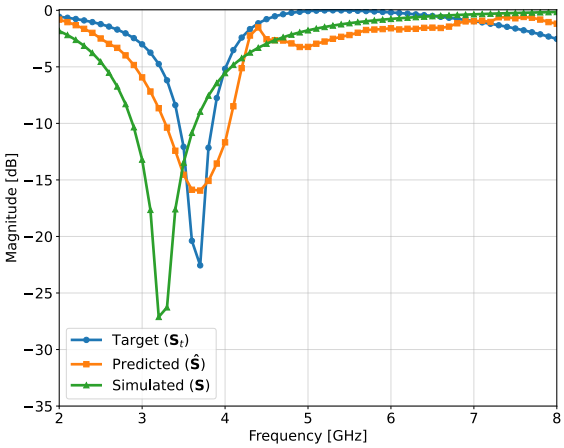


Figure 4.12: Sample 7 - target, predicted, and simulated S-curves

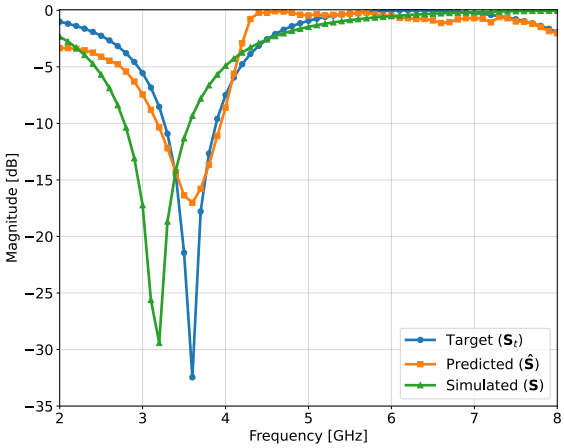


Figure 4.13: Sample 8 - target, predicted, and simulated S-curves

5

Conclusion

In this project, an end-to-end framework for the inverse design of single-layer FSS has been developed. The framework consists of a pattern generator (the decoder part of the cVAE), a surrogate model, and gradient-based latent-space optimization. Although it is not a complete product yet, it can still be used in the early phases to inspire designers with patterns. We have demonstrated a proof of concept that confirms that generative ML is a promising path toward solving the inverse problem of FSS. This framework significantly reduces time and computational resources, as it replaces a trial-and-error workflow. Broadly speaking, it lowers the entry barrier for engineers working on FSS design. Furthermore, the entire methodology is transferable to other inverse design problems, such as designing metasurfaces in mechanics.

5.1 Summary of Research Findings

This thesis demonstrates that generative machine learning can effectively perform the inverse design of FSS unit cell patterns. A dataset of 10,000 simulated $(P, \mathbf{S})$ samples enabled robust training, and the surrogate model achieved mean absolute errors between 0.5 and 1.9 dB. The framework yielded good results for half of the samples and poorer results for the other half, thereby demonstrating its potential. By optimizing the latent space, the method produces designs with S-parameter curves that closely match the desired responses, achieving resonant frequency deviations as low as 0.0-0.2 GHz in half of the benchmarking cases. As shown in Figure 4.6, increasing the dataset size further improves the surrogate model's performance.

5.2 Recommendations for Future Work

5.2.1 Improving the Performance

The first step forward in this application is to improve the model's performance to provide patterns with EM-behavior closer to the desired one. This can be done by simulating more relevant data (patterns that have at least one clear valley in the S-curve), see Figure 4.6. The dataset in this thesis consisted of only 10,000 samples (with only 8,000 used for training), which is a very small dataset size in

terms of machine learning. Another measure is to experiment with other kinds of neural network architectures, especially for the surrogate model to improve its performance. These could be the famous network architectures Xception, VGG16, VGG19, and ResNet50.

5.2.2 Inclusion of Incident Angles and Polarization

In this work, only the boresight condition (0 degrees incident angle) is analyzed for the simulation of S-parameters. If the study is extended to include other incident angles, such as 60 degrees, a more comprehensive understanding of the FSS functioning would be possible. Additionally, the second polarization could be included, as the current study only examines the first. This type of study will require the creation and analysis of a more extensive dataset to accurately capture the changes.

5.2.3 Integration of Multi-Layered FSS

This research has been limited to single-layer FSS in order to manage the complexity of the design process. Future work should consider the extension to multi-layered FSS designs, which offer the potential to achieve more complex frequency responses through coupling effects. Incorporating multi-layered FSS will require a much more extensive dataset.

5.2.4 Material and Dimension Parameterization

In this study, the material choice and the thickness of the dielectric substrate were kept fixed to streamline the design process. A promising direction for future research is to incorporate several common materials, both metals (e.g. copper) and dielectrics (e.g. foam), along with dielectric thickness as adjustable parameters. Allowing the network to select the optimal combination of these parameters could lead to significant improvements in performance, though this approach will demand a substantial increase in simulated data to cover the expanded search space.

5.2.5 Alternative Pattern Representation Techniques

The current approach uses a rasterized image (128×128 pixels) representation for FSS patterns, which may impose limitations on the resolution and the fidelity of geometrical details. Future studies could explore the use of higher resolution methods or alternative representation techniques, such as vector-based formats (e.g., SVG), to mitigate issues associated with rasterization. Such enhancements are expected to provide smoother and more precise pattern reconstructions.

Bibliography

[1] Ben A. Munk. *Frequency Selective Surfaces: Theory and Design*. John Wiley & Sons, Inc., New York, 2000.

[2] Waseem Afzal, Muhammad Zeeshan Baig, Amir Ebrahimi, Md. Rokunuzzaman Robel, Muhammad Tausif Afzal Rana, and Wayne Rowe. Frequency selective surfaces: Design, analysis, and applications. *Telecom*, 5(4):1102–1128, 2024.

[3] Rana Sadaf Anwar, Lingfeng Mao, and Huansheng Ning. Frequency selective surfaces: A review. *Applied Sciences*, 8(9), 2018.

[4] Xingwei Wang, Chen Zhao, Chuanpeng Li, Yu Liu, Shuang Sun, Qiangliang Yu, Bo Yu, Meirong Cai, and Feng Zhou. Progress in mxene-based materials for microwave absorption. *Journal of Materials Science & Technology*, 180:207–225, 2024.

[5] Zhao Zhou, Zhaohui Wei, Jian Ren, Yingzeng Yin, Gert Frølund Pedersen, and Ming Shen. Representation learning-driven fully automated framework for the inverse design of frequency-selective surfaces. *IEEE Transactions on Microwave Theory and Techniques*, 71(6):2409–2421, 2023.

[6] J.N. Hwang, C.H. Chan, and II. Marks, R.J. Frequency selective surface design based on iterative inversion of neural networks. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 39–44 vol.1, 1990.

[7] Bora Döken and Mesut Kartal. Easily optimizable dual-band frequency-selective surface design. *IEEE Antennas and Wireless Propagation Letters*, 16:2979–2982, 2017.

[8] Pekka Alitalo and Sergei Tretyakov. Electromagnetic cloaking with metamaterials. *Materials Today*, 12(3):22–29, 2009.

[9] J. B. Pendry, D. Schurig, and D. R. Smith. Controlling electromagnetic fields. *Science*, 312(5781):1780–1782, 2006.

[10] Komal Kaur and Amanpreet Kaur. Frequency selective surfaces (fss) for s and x band shielding in electromagnetic applications. *AIP Conference Proceedings*, 2576:030015, 12 2022.

[11] Xuehan Chen, Jingjing Tan, Litian Kang, Fengxiao Tang, Ming Zhao, and Nei Kato. Frequency selective surface toward 6g communication systems: A contemporary survey. *IEEE Communications Surveys & Tutorials*, 26(3):1635–1675, 2024.

[12] Leidiane CMM Fontoura, Hertz Wilton De Castro Lins, Arthur S Bertuleza, Adaildo Gomes D'assunção, and Alfredo Gomes Neto. Synthesis of multiband frequency selective surfaces using machine learning with the decision tree algorithm. *IEEE Access*, 9:85785–85794, 2021.

[13] Varun Chaudhary and Ravi Panwar. Machine learning empowered magnetic substrate coupled broadband and miniaturized frequency selective surface. *IEEE Transactions on Electromagnetic Compatibility*, 65(2):406–413, 2023.

[14] Varun Chaudhary and Ravi Panwar. Machine learning derived tio 2 embedded frequency selective surface for emi shielding applications. *IEEE Transactions on Dielectrics and Electrical Insulation*, 30(5):2205–2212, 2023.

[15] Riqiu Cong, Ning Liu, Xiao Li, Hongwei Wang, and Xianjun Sheng. Design of wideband frequency selective surface based on the combination of the equivalent circuit model and deep learning. *IEEE Antennas and Wireless Propagation Letters*, 22(9):2110–2114, 2023.

[16] Yaxi Pan, Jian Dong, Meng Wang, Heng Luo, and Yadgar I Abdulkarim. Inverse design of ultra-wideband transparent frequency selective surface absorbers based on evolutionary deep learning. *Journal of Physics D: Applied Physics*, 56(41):415002, 2023.

[17] Li-Ye Xiao, Yu Cheng, Yan-Fang Liu, Fu-Long Jin, and Qing Huo Liu. An inverse topological design method (itdm) based on machine learning for frequency selective surface (fss) structures. *IEEE Transactions on Antennas and Propagation*, 2023.

[18] Parinaz Naseri and Sean V Hum. A generative machine learning-based approach for inverse design of multilayer metasurfaces. *IEEE Transactions on Antennas and Propagation*, 69(9):5725–5739, 2021.

[19] C. A. Balanis. *Antenna Theory: Analysis and Design*. Wiley, 4th edition, 2016.

[20] D. M. Pozar. *Microwave Engineering*. Wiley, 4th edition, 2012.

[21] Samuel Harrison. *Exploring and Exploiting Charge-Carrier Confinement in Semiconductor Nanostructures: Heterodimensionality in Sub-Monolayer InAs in GaAs and Photoelectrolysis Using Type-II Heterojunctions*. PhD thesis, Lancaster University, 2016. PhD Thesis. Accessed: Mars 2025.

[22] R. F. Harrington. *Time-Harmonic Electromagnetic Fields*. IEEE Press, 2001.

[23] David M. Lovinger. Communication networks in the brain: Neurons, receptors, neurotransmitters, and alcohol. *Alcohol Research & Health*, 31(3):196–214, 2008.

[24] Bernhard Mehlig. *Machine Learning with Neural Networks: An Introduction for Scientists and Engineers*. Cambridge University Press, 2021.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS 2017)*, pages 5998–6008, 2017.

[27] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014.

[28] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3483–3491, 2015.

[29] Institute of Electrical and Electronics Engineers (IEEE). Ieee standard letter designations for radar-frequency bands. *IEEE Std 521-2019 (Revision of IEEE Std 521-2002)*, pages 1–15, 2020.

[30] Rogers Corporation. RT-duroid 5870 - 5880 Data Sheet, August 2022. Accessed: 2025-03-05.

[31] Edward A. Parker. *The Gentleman's Guide to Frequency Selective Surfaces*. Kent, UK, April 1991. First presented at the 17th Q.M.W. Antenna Symposium, London, April 1991.

A

Appendix

A.1 Target, predicted, and simulated *S*-curves for benchmarking samples

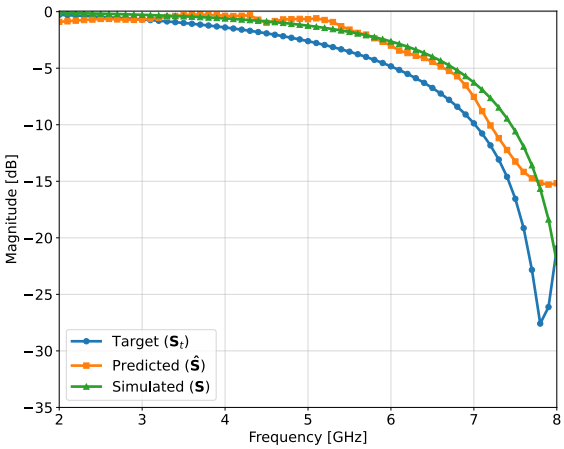


Figure A.1: Sample 1 - *S*-curves

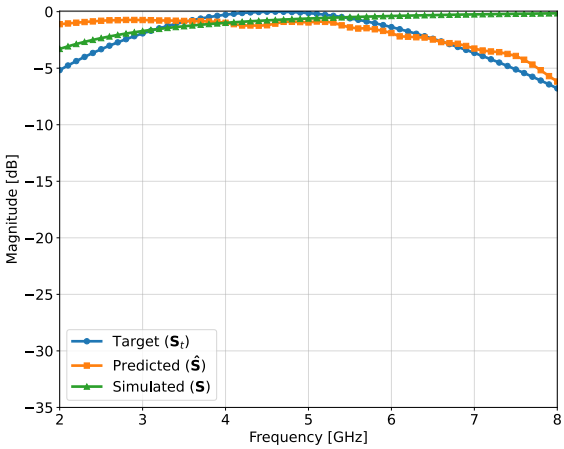


Figure A.2: Sample 2 - *S*-curves

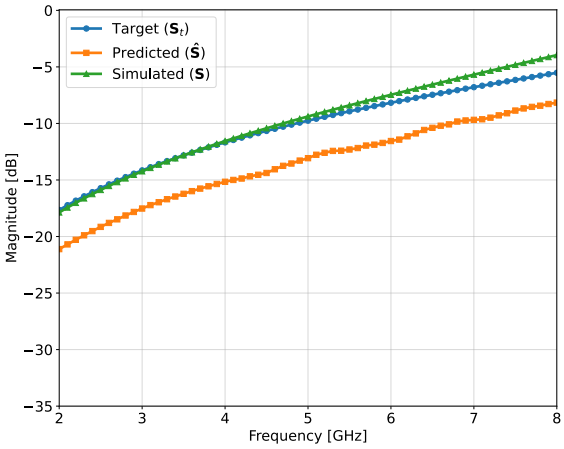


Figure A.3: Sample 3 - *S*-curves

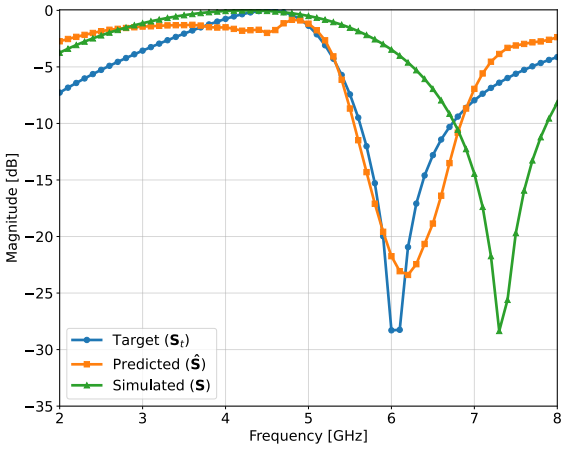


Figure A.4: Sample 4 - *S*-curves

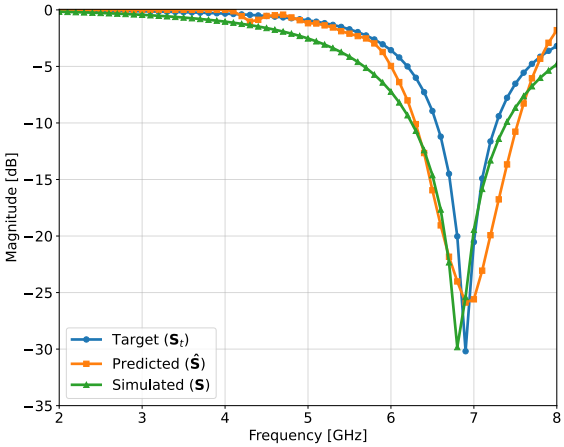


Figure A.5: Sample 5 - S-curves

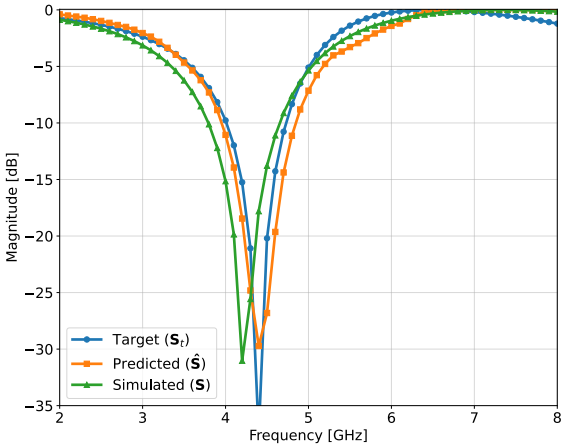


Figure A.6: Sample 6 - S-curves

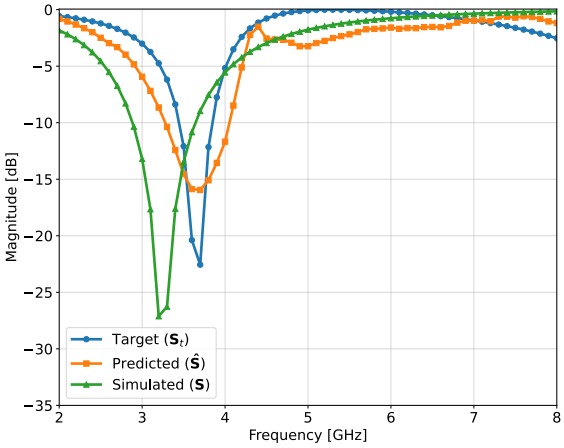


Figure A.7: Sample 7 - S-curves

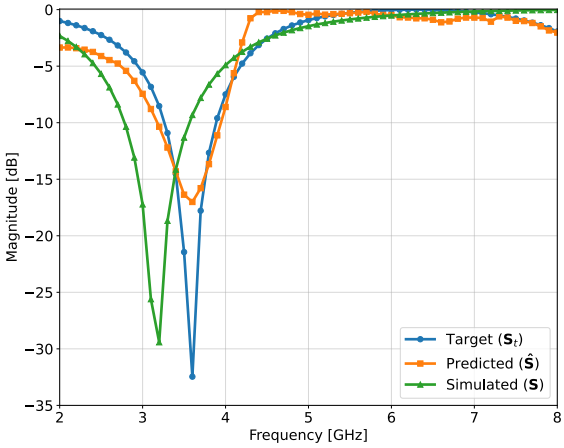


Figure A.8: Sample 8 - S-curves

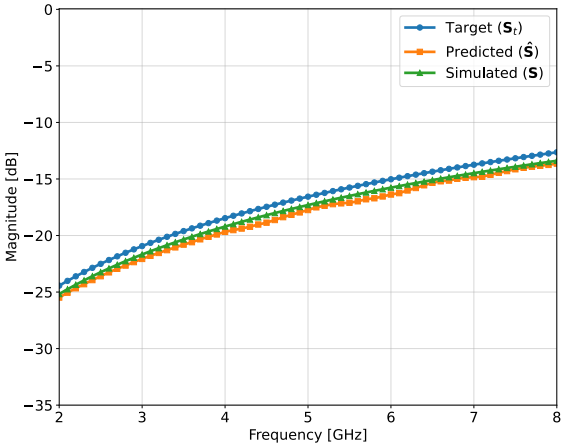


Figure A.9: Sample 9 - S-curves

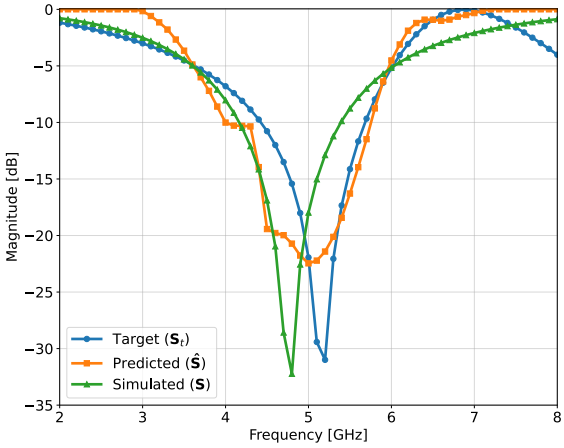


Figure A.10: Sample 10 - S-curves

A.2 Optimized unit cell pattern for benchmarking samples

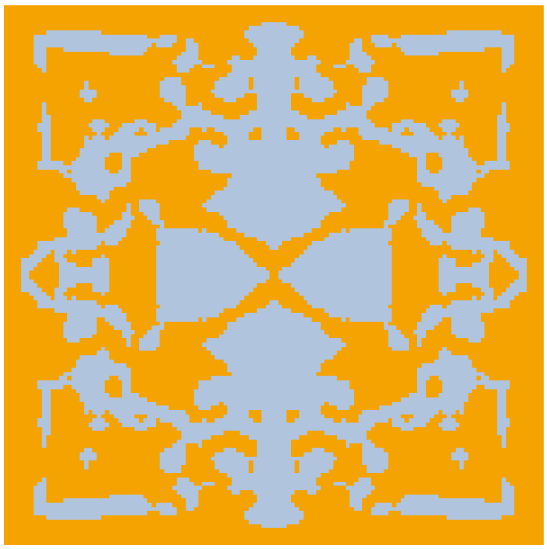


Figure A.11: Sample 1 - optimized pattern

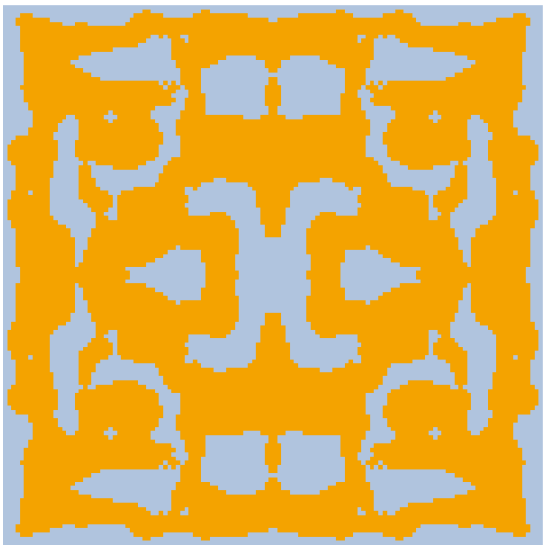


Figure A.12: Sample 2 - optimized pattern

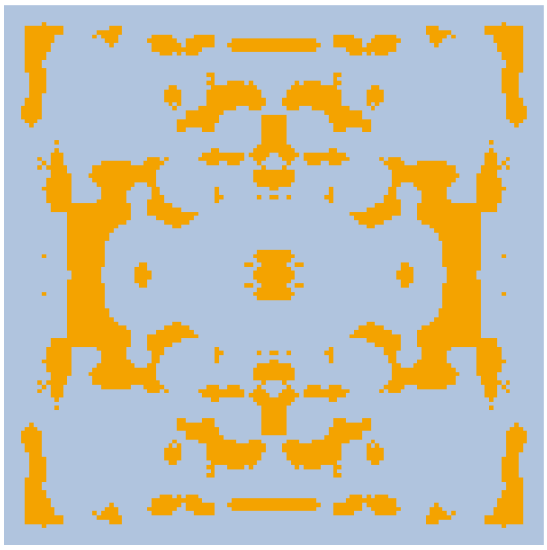


Figure A.13: Sample 3 - optimized pattern

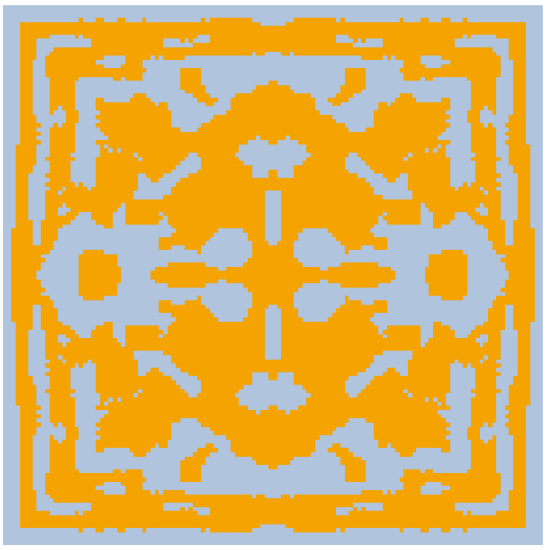


Figure A.14: Sample 4 - optimized pattern

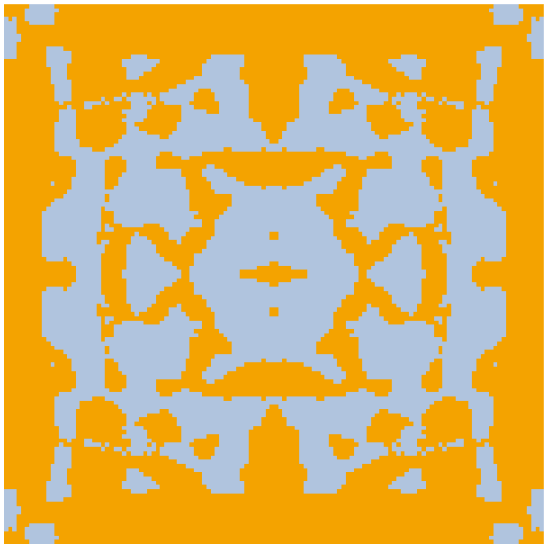


Figure A.15: Sample 5 - optimized pattern

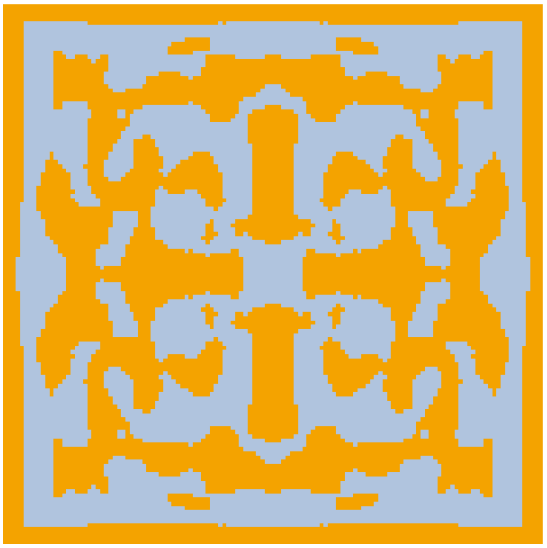


Figure A.16: Sample 6 - optimized pattern

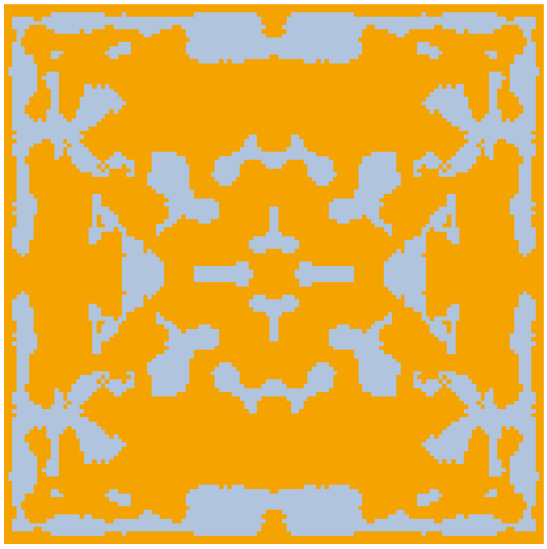


Figure A.17: Sample 7 - optimized pattern

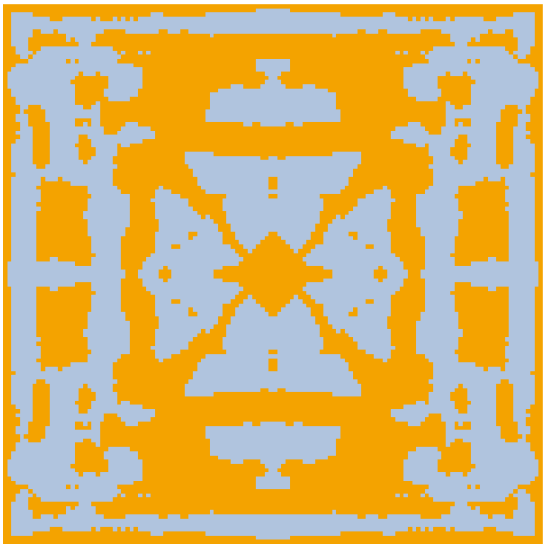


Figure A.18: Sample 8 - optimized pattern

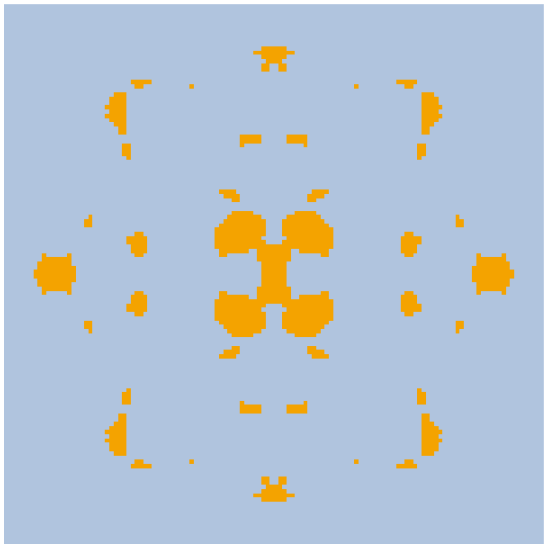


Figure A.19: Sample 9 - optimized pattern

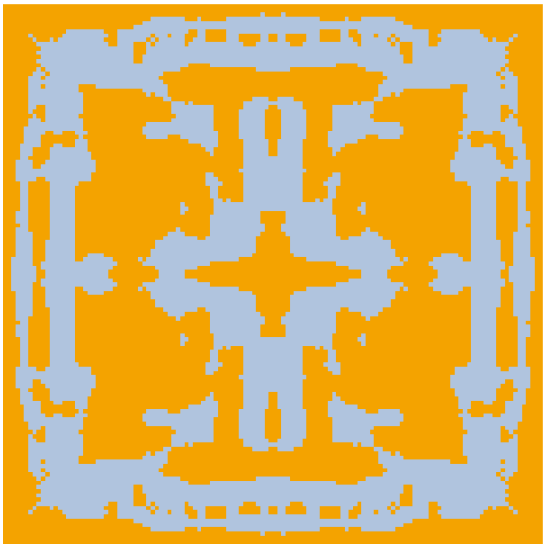


Figure A.20: Sample 10 - optimized pattern

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

CHALMERS
UNIVERSITY OF TECHNOLOGY