Erasmus+

Erasmus Mundus Joint Master degree
Photonics for Security Reliability and Safety

Master *Photonics for Security Reliability and Safety (PSRS)*

A DEEP LEARNING APPROACH FOR PREDICTING TRANSMISSION SPECTRA OF METASURFACES

Master Thesis Report

Presented by

Md Imran Hossain

and defended at

University Jean Monnet

04 September 2024

Academic Supervisor: Professor, Carlo Ricciardi
Host Supervisor: Professor, Humeyra Caglayan

Jury Committee:
Professor, Nathalie Destouches, University Jean Monnet
Professor, Matthieu Roussey, University of Eastern Finland
Professor, Jean-Philippe Colombier, University Jean Monnet

A DEEP LEARNING APPROACH FOR PREDICTING TRANSMISSION SPECTRA OF METASURFACES

Abstract

Metasurfaces, ultrathin 2D structures, have drawn considerable attention in recent years due to their applications in many fields. The metasurface is a subwavelength-engineered structure with unique electromagnetic properties and exceptional light-matter interactions. Recent research indicates that metasurfaces are a good fit for studying the polarization conversion of electromagnetic waves. In search of polarization conversion to any arbitrary angle, we have considered taking advantage of deep learning to investigate the field thoroughly. In recent years, the contribution of deep learning in optics and photonics has been undeniable. This work presents a neural network miming a conventional numerical simulator like Ansys Lumerical FDTD. We have introduced a neural network based on ResNet-18 to predict the transmission spectra of Gold metasurfaces within the 1200 to 1700 nm region. The model is trained with thousands of simulated data of metasurfaces with varied geometries. The model can accurately predict the transmission spectra within a couple of milliseconds. Evaluation of 150 datasets shows that the model has an average prediction accuracy of 85.27%. The model aims to present a time-efficient method for investigating polarization conversion metasurfaces.

Acknowledgement

I want to express my deepest gratitude to all those who have supported and guided me throughout my EMJMD PSRS journey.

First and foremost, I am deeply grateful to my supervisor, Prof. Humeyra Caglayan, for allowing me to be a part of the team Metaplasmonics and work on an exciting research idea. Your invaluable guidance, encouragement, patience throughout this research, and expertise have been instrumental in shaping my work, and I am thankful for the opportunities you have provided me. I also owe a debt of gratitude to my mentor and colleague Linzhi Yu, one of the brightest individuals in the Metaplasmonics group, for your unceasing advice, perceptive conversations, helpful criticism, and recommendations that enhance both the caliber of the work and my understanding. I am also very grateful to my colleagues, whose friendship and enlightening conversations during group meetings enhanced the quality of the research. It's like a family to me, and I'm thankful to everyone for their unwavering personal and academic support during this journey.

To my friends and family, especially my parents, my beloved wife Tanha, and my closest friends Sydur and Shuvo, I am forever indebted to you for your unwavering love, support, and belief in me, even during the most challenging times. Your encouragement has been my strength throughout this journey.

Thank you all for being part of this incredible journey. This achievement would not have been possible without you.

Contents

List of Figures

List of Tables

1 Introduction

Polarization is a fundamental characteristic of light. The orientation of the oscillating electric field of an electromagnetic wave describes the polarization of light. Due to its intrinsic characteristics, polarization of light is extensively explored in integrated optics [1], light-matter interaction [2], imaging [3], quantum optics[4], nonlinear optics [5], wireless communication[6], and sensing applications [7]. Precise control and manipulation of polarization are important in many optics and photonics devices since they have a high degree of freedom when controlling electromagnetic waves. A perfect polarization converter can transform an electromagnetic wave with an unspecified polarization into a clearly defined one. Conventional instruments such as wave plates and polarizers handle the polarization of light as though it is constant throughout the beam rather than concentrating on particular segments; they collectively modify the entire beam's polarization. This simplifies polarization management by treating the light beam as a single, homogeneous unit for large-scale modifications [8]. Birefringent materials like calcite are commonly used to make traditional wave retarders. These materials have different refractive indices for optical transmission, which results in a phase difference between two perpendicular axes. Unfortunately, this technique typically requires an optical channel long enough to collect the phase difference, leading to large, unsuitable wave retarders for advanced applications that call for small optical elements. Furthermore, material limitations limit the operational wavelength range of conventional wave retarders, impeding the development of contemporary optical technologies [9].

Considering the above fact, metasurface can be a promising candidate for developing polarization rotators. Metasurfaces consist of subwavelength resonance units arranged in a two-dimensional planar structure. By adjusting the characteristics of these units, such as geometry and size, metasurfaces can control how light interacts with them, enabling precise modulation of electromagnetic wave properties like amplitude, polarization, and phase [10]. Despite having the ability to possess desired light-matter interaction, we have not been able to take full advantage of metasurfaces. A perfectly engineered metasurface can produce various optical and electromagnetic functionalities over any frequency spectrum. However, such a well-designed metasurface is not easy to achieve. Predicting a material's optical characteristics and approximate structure involves sophisticated repetitive computations because the generalized theory cannot resolve the complex physical principles that characterize these light-matter interactions at the nanoscale [11]. Designing these metasurfaces requires knowledge of handling electromagnetic simulation software and an understanding of electromagnetic optics. In addition, human error is a natural part of the classic metasurface design process. Also, it demands an extensive parameters sweep of the geometry and material properties based on a trial and error method that takes hours of simulations.

Implementing machine learning in optics and photonics has recently drawn considerable attention and has been labeled AI in photonics. Metasurfaces are also not excluded from the coordination of Machine Learning, especially Deep Learning. It is possible to predict the geometry of the metasurfaces and their corresponding electromagnetic features without extensive computation of Maxwell's equations by using deep neural networks[12, 13]. These frameworks could be a potential approach to investigate the polarization conversion metasurface.

1.1 Polarization of Lights

As it is known that polarization of light refers to the orientation of the electric field based on the amplitude and phase distribution of the electric field between two orthogonal directions, the polarization of light can be classified into several types, such as

- Linearly polarized light
- Circularly polarized light
- Elliptically polarized light

To understand the polarization closely, let's consider a plane wave propagating in the z-direction at the speed of light, considering the electric field-

$$\vec{E} = \vec{E_0} e^{i(\vec{k}\cdot\vec{z}-\omega t)} \tag{1.1}$$

As $\vec{E_0}$ is orthogonal to the propagation direction so, $\vec{E_0} = (E_x, E_y, 0)$ and complex amplitude E_x and E_y has both magnitude and phase. Thus results in writing $E_x = |E_x|e^{i\phi_x}$, $E_y = |E_y|e^{i\phi_y}$.

Firstly, the plane wave can be considered linearly polarized if the phase difference between E_x and E_y becomes zero. If any of the orthogonal components of the electric field become zero, this can still be identified as linearly polarized but in a specific direction. For instance, if $E_y = 0, E_x \neq 0$ then $\vec{E} = E_0 \hat{x} e^{i(k\cdot z-\omega t)}$, this can be called linear polarized in the x direction (x-polarized). Similarly, $E_x = 0, E_y \neq 0$ then $\vec{E} = E_0 \hat{y} e^{i(k\cdot z-\omega t)}$, this can be called linearly polarized in the y direction (y-polarized). For better understanding, x-polarized and y-polarized light is illustrated in Figure 1.1[14].

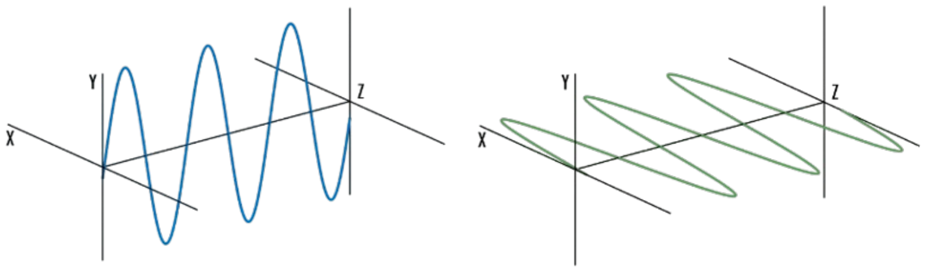


Figure 1.1: Orientation of the electric field for the x-polarized(right) and y-polarized(left) light.

Secondly, The plane wave can be considered as circularly polarized if the magnitude of the electric field's orthogonal components become equal ($E_x = E_y$) and constantly maintains a quarter wavelength out of phase ($\phi_x - \phi_y = \frac{2\pi}{\lambda}\frac{\lambda}{4}$). During the propagation along the z-direction, if the phase difference is $\frac{\pi}{2}$, the polarization(E) rotates clockwise in the $x - y$ plane, which is called Left Circularly Polarized (LCP). Similarly, if the phase difference is $-\frac{\pi}{2}$, the polarization (E) rotates counterclockwise in the $x-y$ plane, which is called Right Circularly Polarized (RCP). LCP and RCP light is shown in Figure 1.2[15] to visualize properly.

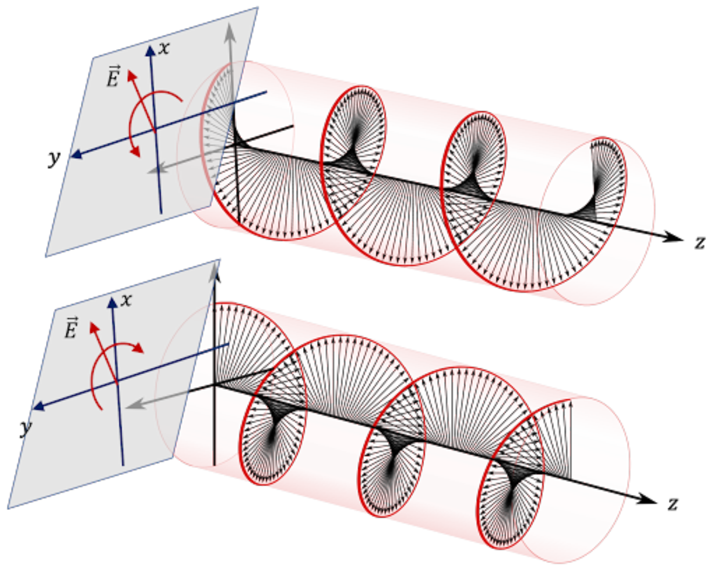


Figure 1.2: Orientation of the electric field for the circularly polarized light.

Lastly, if the E_x and E_y are not equal in magnitude and possess a phase difference($\Delta\phi \neq 0 \neq \frac{\pi}{2}$), the magnitude of the polarization(E) varies as it rotates within $x-y$ plane, from the z-direction the orientation of the changing electric field resulting in ellipse, so this is called elliptical polarization. In Figure 1.3[14], elliptical polarization is illustrated for better conceptualization.

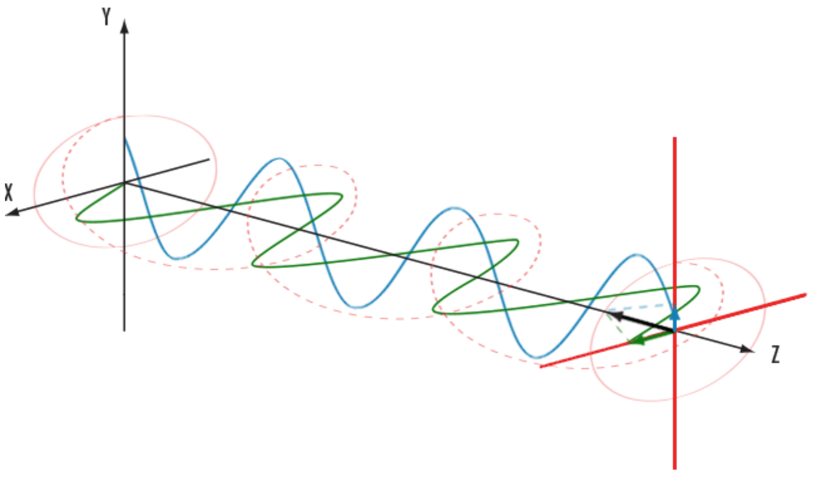


Figure 1.3: Orientation of the electric field for the elliptically polarized light.

1.2 Physics and Applications of Metasurface

A metasurface can be considered a bulk metamaterials' two-dimensional (2D) counterpart. Metasurfaces, or planar metamaterials with subwavelength thickness, comprise stacks of single or few layers of planar structures. Metamaterials are synthetic materials designed to possess characteristics not present in ordinary materials. Rather than their content, the structure of these materials gives them their distinctive qualities. Metamaterials are three-dimensional (3D) structures. They are typically composed of repeating unit cells smaller than the wavelength of the electromagnetic waves they interact with. Metamaterial exhibits exceptional electromagnetic properties, and their permittivity and permeability are derived from their structure orientation rather than the material itself[16], artificial magnetism, negative permittivity, and negative permeability have been noted in fabricated metamaterials along with negative refractive index[17]. Russian physicist Victor Veselago first proposed the idea of metamaterials in 1968. Many typical electromagnetic properties could be reversed by materials with a negative refractive index, according to Veselago's theory[18]. Early in the new millennium, British scientist Sir John Pendry contributed to the field's development of the theoretical framework and practical designs for metamaterials. Specifically, Pendry and his associates presented the notion of achieving negative permeability and negative permittivity by artificially structured materials, including split-ring resonators, which resulted in the creation of metamaterials with negative refractive indices[19]. From then until today, metamaterial has been used in assorted engineering applications counting perfect lens [20], invisible cloak[21], antenna[22], sensor [23], holography [24], absorber[25], etc. However, the large losses, considerable dispersion related to resonant responses, and difficulties in fabricating nanoscale 3D structures pose significant challenges to the practical implementation of metamaterials and the need to ensure the structure is bulky enough to aid essential properties like refractive index, permeability, and permittivity [26]. Meanwhile, the effective permittivity, permeability, and refractive index in metasurfaces are less significant since the subwavelength thickness adds a small propagation phase.

On the other hand, the surface or interface reflection and transmission, together with their amplitude, phase, and polarization states that come from the customized surface impedance, are greatly influential[27]. Metasurfaces can be easily fabricated by lithography and nanoprinting techniques, including stacking single or few planar structures. The extremely thin thickness in the direction of wave propagation can significantly reduce undesired losses with the help of appropriate material and metasurface structures. Metasurfaces have a wide range of fruitful applications including flat optics [28], beam steering [29], holography[30], polarization control[31], wavefront shaping and manipulation[32], sensing and imaging[33], etc. All things considered, metasurfaces can get beyond bulk metamaterials while having strong enough interactions with incident waves to produce exceptionally useful characteristics.

1.3 Deep Learning in Optics and Photonics

In the past few decades, novel approaches to manipulating light-matter interactions have been made possible by developing constructed photonic structures, including plasmonic nanostructures, metamaterials, and photonic crystals. These developments have made a wider range of applications and new device designs possible. Structural designs are crucial whether we are talking about single plasmonic nanostructures, metamaterials, or photonic crystals made of metallic or dielectric building block arrays. Traditionally, numerical simulation techniques or physics-based approaches have governed the design of photonic structures[34]. We must frequently adjust the shape and rerun simulations to get closer to the desired responses. This process mainly depends on prior expertise with the design templates; due to time and simulation power limitations, only a few design parameters are changed in finding the ideal structure. However, these methods frequently have trouble with intricate geometries and need a lot of iterations to produce the desired results.

By effectively navigating vast design spaces and identifying counterintuitive solutions, deep learning, a subset of machine learning, provides a data-driven methodology that can supplement conventional techniques. With its sophisticated tools for developing, evaluating, and optimizing optical systems, deep learning has significantly impacted optics and photonics. It has transformed many photonics research fields in recent years, including wavefront shaping, device optimization, material design, and image processing. Deep learning is used in photonics to solve complex design problems. When desired optical properties are specified initially, and the model creates a structure that satisfies those requirements, it is used in the inverse design of photonic structures[35]. Polarization conversion is especially useful for enhancing the function of components like waveguides and photonic crystals for applications like filtering and light trapping. Deep learning also expedites the creation of metasurfaces, which are two-dimensional structures that control light at a subwavelength scale, by forecasting their optical performance in a matter of minutes rather than weeks[36]. In addition, deep learning has brought diversity and more rapid progress in metasurface design and optimization [37], optical image processing and microscopy[38], optical communication and signal processing[39], ultrafast optics and spectroscopy[40], optical material discovery[41] and so on.

Optics and photonics now have exciting new opportunities because of deep learning but also some challenges. One reason is that large amounts of data are typically complex for these models to learn from in highly specialized fields like photonics. Neural networks are also frequently called "black boxes" because it can be challenging to grasp their inner workings and how they connect to the physical phenomena under study[42]. This can make it difficult to understand why they make certain judgments. Also, training these models might take a lot of processing effort, making them resource-intensive. In photonics and optics, deep learning has enormous potential despite these obstacles. We can anticipate seeing increasingly complex optical systems and technologies that are more effective and capable of entirely new things as technology develops. Advances are being made possible by the combination of machine learning and photonics.

1.4 Objective

Our main research goal is to control the polarization of light to any arbitrary angle using fabrication-friendly metasurfaces. The polarization conversion of the transmitted light that passed through the metasurfaces will be investigated in the optical communication wavelength region (1200-1700 nm). We will introduce a deep learning framework that will generate a combination of metasurface unit cells to serve the desired functionalities. A compositional pattern-producing network(CPPN)[43] will be trained with thousands of geometric data with the help of Wasserstein generative adversarial networks (WGAN)[44]. The network will read the pixel coordinates of the input geometries and will predict compositions for those pixels to compose new geometry to gain the design objective. The WGAN will guide the network in composing real images like the input geometries. Since the network will generate hundreds of compositional metasurfaces and it is necessary to check the validity of those generated patterns, numerical simulation is needed to be performed on those metasurfaces. Designing those generated metasurfaces and simulating them with commercially available simulators would take months of labor and require professional expertise. At this level, a framework would be needed to design and simulate those complex structures in short timescales to investigate their light-matter interactions. In this work, we particularly focus on developing a neural network simulator that mimics the behavior of a numerical simulator for a particular design objective. More precisely, a neural network was built to read the metasurfaces and predict the corresponding transmission spectra within the optical communication wavelength region.

2 Background and Literature Review

Neural networks have proven to be effective tools for forecasting intricate physical phenomena as the field of photonics expands. Metasurfaces present a viable avenue for advances in photonics and optics because of their ability to alter light at exceedingly small subwavelength scales. Researchers can anticipate with accuracy how light interacts with these metasurfaces by utilizing deep learning models. This improves our understanding of how electromagnetic waves interact with sophisticated materials and expedites design. In this study, we will cover important research that uses deep learning to anticipate behaviors in optics and photonics, and we will investigate the fundamentals of neural networks, with a specific focus on convolutional neural networks (CNNs).

2.1 Analysis of Recent Studies

An et al. present an accurate yet greatly shortened characterization time-to-resolution metasurface modeling method based on deep learning in their research[45]. They have proposed a CNN-based predicting neural network(PNN) that has the ability to predict transmission spectra for unscripted forms of metasurfaces within 5 μm to 10 μm wavelength regions. Different periodicity, a range of refractive indices, different geometric patterns, and various thicknesses of the metasurface are introduced to the model to make the model applicable to a wide range of metasurfaces. This approach can anticipate the spectral response for a broad range within a short period that outperforms the traditional numerical solver with a great deal of time. This work focuses on predicting not only amplitude but also the phase that allows the researcher to study phase gradient meta-optical devices. They performed the training on PNN with randomly designed 100,000 free-form metasurfaces with randomly associated parameters within a specific range. To generate label datasets, Finite Element Method(FEM) is used to simulate those patterns numerically. The training accuracy of the model was checked based on the MSE loss, which was comparatively low. Though they trained the model with four different parameters, the number of varied geometry was smaller in the datasets. Authors have claimed that the proposed model is easily transferable to different areas of physics where quick and precise modeling is needed to establish a connection between a large as well as complex parametric space and associated physical phenomena.

An interesting deep-learning method is presented by Roberts and Hedayati in their research [46] to predict the structural color of the plasmonic metasurfaces. Frequency-selective absorptions and scattering result in structural color in specially designed metasurfaces. The main idea of this research is to predict the wavelength that is collectively reflected from those metasurfaces that give the appearance of different colors. The model presented in the article can predict the structural color with more than 96% accuracy and dramatically less computational time than a numerical solver like Ansys Lumerical FDTD. Instead of playing with different geometries, this method works on a single geometry with various thicknesses and diameters that cover a wide spatial range in nm. Constant periodicity was used for the metasurfaces in order to preserve viewing angle independence and minimize parameters in the neural network model. The techniques described here offer a generalized way for deep learning-based complicated metasurface design in addition to the evolution of specific metasurface models.

In this article[36], a quick and effective way to model metasurface and map the structure to predict S-parameter is presented, called REACTIVE. Additionally, it demonstrates the benefits of automating the design process to make it more effective, efficient, time and computationally resource-efficient. The average accuracy is recorded at 76.5% while the model works considerably well on 30% datasets. The design approach consists of an 8x8 matrix frame where 1 indicates the existence of copper and 0 indicates the space is blank. CST MWS has been used to simulate the generated pattern to create a set of metasurface and label data to train the REACTIVE model. According to the authors, The advantage of using REACTIVE is that it can automatically create the metasurface structure, freeing engineers to focus on their design goals instead of the other intricate design process.

Complex all-dielectric metasurface modeling is demonstrated by Nadell et al. utilizing deep neural networks, with inputs being the geometry of the metasurface and the underlying physics information[47]. The deep neural network is composed of a series of fully connected layers that are fed by the geometric datasets. The model serves as an optimizer rather than a universal simulator for specific design tasks. The process concerns keeping the metasurface structure similar and varying the height and radius of the cylindrical shapes and lattice period. Within the range of 0.8 to 1.5 THz, the model is able to predict the frequency-dependent transmission spectra precisely.

Lastly, based on deep convolutional neural networks, forward prediction for random metasurfaces operating at terahertz (THz) is presented in this article[48]. The neural network can efficiently predict the amplitude and phase reflected from the metasurfaces within 0.2 to 2 THz. A 5x5 matrix is used to create about 70,000 metasurfaces to train the neural network to predict the amplitude and phase response while keeping the other structural parameters constant. Though the number of metasurfaces is enormous, all the patterns generated look similar to a 5x5 QR code. Over the 10,000 test datasets, the average MSE loss for the amplitude and phase prediction was calculated 0.0011 and 0.0113 respectively. According to the authors, the prediction executes in 3 ms, which is 40,000 quicker than the conventional numerical solver.

2.2 Analysis of Key Findings

Following a thorough review of recent studies using a deep learning-based methodology, we have found that most of the work focuses on amplitude response prediction. Some work focuses on both amplitude and phase prediction in the GHz and THz regions. Different research groups have investigated both the reflection mode and transmission mode thoroughly. We have found some research concerning the optical communication wavelength in this domain. The number of work that focuses on various geometric data rather than parameter optimization is few, and the number of geometries in the datasets lacks diversity in their pattern. The model training and response time in unseen data prediction are also key factors. One of the key issues we have addressed is that the metasurface generation is not convenient with a 5x5 or 8x8 matrix combination, which reduces the degree of freedom in generating random and complex structures. Most of the work is based on CNN and fully connected layers. Improving prediction accuracy can still be done when datasets have a variety of shapes. There is plenty of scope to work on deep learning-based forward prediction of the transmission spectra in optical communication wavelength. Based on recent studies, it is worth mentioning that the deep learning-based approach is more efficient and less time-consuming while maintaining less computational resources.

2.3 Background Study

Convolutional neural networks, or CNNs, are a particular class of artificial neural networks that have gained popularity for their ability to analyze structured input, such as images[49]. CNNs are flexible and can be used for a variety of other data analysis and classification applications, even though they are primarily recognized for image processing. Consider CNNs to be a specific type of Artificial Neural Network (ANN) built to identify patterns in visual data, textures, or even the spatial relationships between such patterns. CNNs are very good at jobs like image analysis because of their capacity to recognize complex patterns. However, their usefulness goes beyond just images; they are also extensively employed in natural language processing[50], medical imaging[51], and video recognition[52]. CNNs process incoming data by sending it via multiple hidden layers and then fully linked layers, enabling the network to convert the input into useful outputs. A CNN turns out to be a strong and effective method for handling this type of image data processing, considering our requirement to examine 2D metasurface images and extract intricate information in order to forecast transmission spectra.

2.3.1 Convolutional Layer

The core component of a CNN is the convolutional layer, which uses the convolution process to identify local patterns in the data. This layer generates feature maps by scanning the input data using tiny filters, or kernels. These maps draw attention to significant details such as forms, textures, and edges[53]. In essence, the kernels are tiny matrices that traverse the data, carrying out calculations at every location. The stride determines the amount of movement the kernel makes each time. These kernels typically come in 3x3, 5x5, or 7x7 sizes, and the edges of the image are frequently padded as they travel over the data grid. This guarantees that no details are lost at the borders and that the kernels can adequately cover the whole input region. The procedure yields a feature map that effectively captures the salient features of the input data.

2.3.2 Activation Function

Neural networks require activation functions to introduce non-linearity, which enables the networks to recognize and interpret complex patterns in the input. These functions allow the network to simulate non-linear relationships when applied element-wise to the outputs of convolutional layers. The sigmoid, tanh, and ReLU activation functions are frequently found in CNNs. Among these, the Rectified Linear Unit (ReLU) is notably well-liked for CNNs in deep learning. Positive inputs are processed without modification, whereas negative inputs are set to zero. ReLU is a highly effective method for training deep networks because of its simplicity, which helps prevent the vanishing gradient problem. On the other hand, it can occasionally result in the “dying ReLU” problem, in which neurons stop responding to negative inputs. As its output can be understood as probabilities, the sigmoid function, on the other hand, compresses input values into a range between 0 and 1. This makes it ideal for tasks like binary classification. Despite this, the training process may be slowed down by the sigmoid’s propensity for vanishing gradients, particularly for inputs that are far from zero. This is improved by the hyperbolic tangent (tanh) function, which centers the output around zero and squashes inputs into a range between -1 and 1. This frequently results in faster convergence during training. Tanh is vulnerable to the vanishing gradient issue, just like sigmoid. Tanh is a good option for the output layers since it naturally falls inside the complex number range that our design operates with, which is -1 to +1[54].

2.3.3 Pooling Layer

CNNs require pooling layers in order to minimize the spatial size of feature maps. This reduces the number of parameters and computational load. In addition to increasing the model's efficiency, this helps avoid overfitting. Furthermore, pooling layers confer a trait called translational invariance, which makes the network more adept at identifying features wherever they occur in the input[53]. Max pooling and average pooling are the two commonly used pooling strategies. Max pooling reduces the size of the data while highlighting the most noticeable characteristics by choosing the highest value within a narrow sliding window across the feature map. In doing so, dimensionality is decreased, yet important information is retained. The feature map is represented more broadly and smoothly when average pooling is used to calculate the mean value within the window. Both the degree of downsampling and the resolution of the final pooled feature map are influenced by the stride and window size parameters, which govern this process and control how the pooling window moves across the feature map.

2.3.4 Batch Normalization

In deep learning, batch normalization is a technique that scales and modifies the activations of a preceding layer's output to normalize it. Neural network training is greatly accelerated and stabilized by this approach. Normalization involves dividing the result by the batch standard deviation after deducting the batch mean from the activations. In order to standardize the input for the following layer, this step makes sure that the activations have a mean of zero and a standard deviation of one. Following normalization, the data is subjected to learned scaling and shifting parameters, which preserve the network's ability to depict intricate patterns in spite of the normalization. Batch normalization has significant advantages; it lessens internal covariate shift or the variation in the network activation distribution brought about by parameter adjustments. Higher learning rates can be used as a result of this reduction, which may accelerate convergence[55]. Furthermore, batch normalization serves as a regularizer, which frequently eliminates the need for further regularization methods like dropout. Batch normalization facilitates the construction of more resilient and effective neural networks by stabilizing the training process and allowing for higher learning rates.

2.3.5 Fully Connected Layer

Dense layers, also known as fully connected layers, are crucial components of neural networks, particularly CNNs, which are usually positioned near the conclusion of the architecture. These layers create a web of connections by connecting each neuron in one layer to each neuron in the next. In order to help the network produce its final predictions, this structure enables the network to integrate and combine the complicated characteristics that the prior convolutional and pooling layers had taught it. Each neuron in a fully connected layer has a unique set of weights and a bias, which are modified during training in response to variations between the target and the expected output[53]. Backpropagation, which teaches the network to minimize errors, facilitates this correction. These neurons' outputs are subjected to non-linear activation functions, such as tanh, sigmoid, or ReLU, to aid in the model's learning of intricate patterns in the data. The last fully connected layer in classification tasks often generates a set of scores, each of which indicates the likelihood that the input is a member of a particular class.

2.3.6 Forward Propagation

In neural networks, forward propagation is the process by which input data travels through the network layer by layer to produce an output. This procedure is necessary for both estimating the loss during training as well as for making forecasts. The input layer is the first layer that receives and transfers raw data, such as text, photos, or numbers, to the subsequent layer. Every layer of the network changes the data as it moves across it. Filters extract significant information from images, like edges and textures, using convolutional layers. Thus, the network can capture more intricate patterns by introducing non-linearity by applying activation functions. The next step in pooling layers is to reduce the amount of data, which reduces computing overhead and lessens the sensitivity of the model to slight changes in the input. The data is gradually refined by repeating this process with every layer until it reaches the output layer. This is the last prediction the network makes; for classification tasks, this might be a list of probabilities that indicate the likelihood of each potential class.

2.3.7 Backpropagation

An essential component of neural network training is backpropagation, which modifies the model's weights and biases in response to output errors. The first step in the procedure is to compute the loss, which is the difference between the target values and the network's anticipated output. Different loss functions are used depending on the job, such as Cross-Entropy Loss for classification and Mean Squared Error for regression. This loss provides a quick overview of the network's performance. The gradients, which indicate the relative contributions of each weight and bias to the mistake, are then computed by backpropagation. Backpropagation computes these gradients for each layer by going backward through the network and applying the chain rule. In essence, the gradients aid in determining the amount and direction of change that each parameter should undergo in order to minimize the loss. Weights and biases in the network are updated using this data by optimization techniques such as Adam, RMSProp, or stochastic gradient descent (SGD). This process is done numerous times, enabling the network to increase its accuracy as it learns from its errors progressively. In this sense, backpropagation powers neural network learning by continuously improving the model's internal parameters to comprehend and forecast the data it has been trained on.

2.3.8 Loss Function

The difference between the expected output and the target values is measured quantitatively using a loss function. The loss function ensures that the network's predictions get more accurate by directing the optimization process during training by offering an objective criterion to minimize. Cross-entropy loss and Mean Squared Error (MSE) are two of the most widely used loss functions. For regression tasks, where the objective is to predict continuous values, MSE is commonly utilized. MSE calculates the average of the squared differences between the predicted values ($\hat{E}$) and the actual values (E), as described by the equation 2.1.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(E_i - \hat{E}_i)^2 \tag{2.1}$$

In this formula, n represents the number of data points, E_i denotes the true values, and $\hat{E}_i$ stands for the predicted values. By squaring the differences, MSE ensures that larger errors are penalized more heavily, which drives the network to improve its predictions. Once the loss is computed, backpropagation is employed to minimize it. During backpropagation, the gradients of the loss function concerning each network parameter are calculated using the chain rule. These gradients indicate the direction and magnitude of the change needed for each parameter to reduce the loss.

2.3.9 Optimizer

Optimizers are essential to neural network training because they adjust the network's parameters to minimize the loss function and enhance the performance of the model. AdamW is a well-liked optimizer that stands out for its reliability and effectiveness. By separating weight decay from gradient updates, AdamW enhances regularization by improving upon the traditional Adam optimizer, which modifies the learning rate for every parameter based on the first and second moments of gradients. Better management of overfitting is made possible by this, resulting in more stable training. For any optimizer, the learning rate is critical; too high or too low might cause instability in the training process and poor convergence. The capacity of AdamW to modify the rate of learning for every parameter guarantees a more rapid and effective convergence. Other optimizers, such as RMSProp, function well for problems involving recurrent neural networks because they modify the learning rate in response to recent gradient magnitudes. However, SGD adds noise to its parameter updates and updates them using a fixed or decaying learning rate. This can aid in faster convergence, but it also necessitates careful adjustment of the learning rate to prevent divergence or sluggish progress[56].

3 Methodology

The goal of our work is to build a neural network simulator to mimic the numerical simulator like FEM (Finite Element Method), FIT (Finite Integration Technique), and FDTD (Finite-Difference Time-Domain). Indeed, it is not practical to mimic the whole simulator, which has thousands of associated parameters. Our target is to predict transmission spectra (complex electric field) of the Gold metasurface for specific wavelength regions. The purpose of this neural network simulator is to predict transmission spectra of specific metasurfaces within a short period, whereas numerical simulators take hours to simulate. This approach will lead us to a time-convenient solution for other related research domains. The approach is as follows: a metasurface with a specific design will be passed through a neural network, resulting in transmission spectra of that metasurface within a couple of milliseconds. The overall process is illustrated in Figure 3.1.

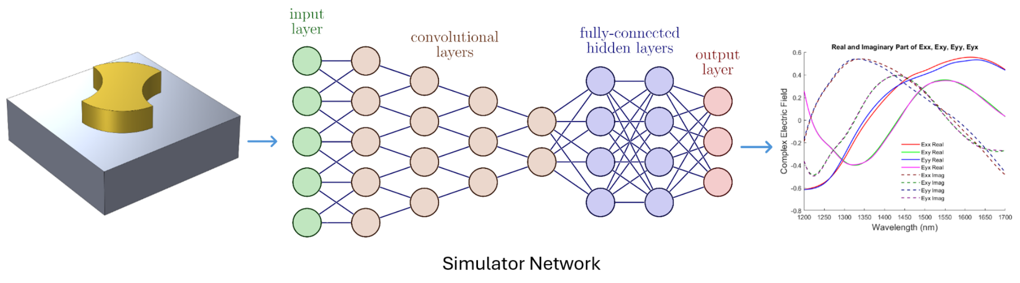


Figure 3.1: Process flow to predict transmission spectra of a metasurface through a neural network.

The main concern with any neural network is how the feature data and label data are introduced to the model. In our case, we wanted to predict the transmission spectra of 2D metasurfaces.

Before beginning the process, it is better to address all the parameters we are working with. The dimension and material properties of the substrate where the metasurface will be placed are fixed for all the metasurfaces, so we do not need to consider the substrate parameters. The metasurfaces are made of Gold with a fixed thickness. So, we don't need to be concerned about the metasurface's material properties and thickness. As we change the shape or pattern of the metasurface thousands of times, we may think about the metasurface as a 2D image. In short, a 2D image of a metasurface will be passed through a neural network, predicting the transmission spectra of the metasurface associated with the 2D image. Generally, training a neural network is related to thousands of training data that match the real case scenario. So, we also need training datasets containing various metasurface and their simulated transmission spectra. The details of the training datasets will be discussed in the following section.

It is well known that a Convolution Neural Network(CNN) is a good image classifier[49]. CNNs are configured to exploit the spatial structure of the images. It uses convolutional layers to learn image features, including patterns, edges, and texture. It can read the specific features from the image and make decisions accordingly by coordinating the label data. Although CNN is superior to image classification, it faces some issues like vanishing gradients, and slow learning, also deeper network has the risk of overfitting. In this work, instead of using CNN, we will use Residual Network(ResNet), a special type of CNN that addresses the issues that CNN can not resolve properly.

3.1 Residual Network (ResNet)

Residual Network(ResNet) is a special Convolutional Neural Network (CNN) that treats vanishing gradient problems effectively. In ResNet, information passes through the residual blocks, which consist of the convolutional layer, activation function, and pooling layer. Generally, when a neural network goes deeper, the gradients become very small during the backpropagation due to the passage of numerous layers[57]. The term here, the gradient becomes so small after passing multiple layers, is called the vanishing gradient. What is the gradient? The gradient is the derivative of the loss function concerning the network parameters. Network parameters refer to weights and biases of the network where weights are associated with the connections between neurons in the adjacent layers, and bias is associated with each of the neurons. This vanishing gradient causes issues with the model learning process, where the model learns very slowly or, in severe cases, doesn't learn at all. This vanishing gradient leads to issues like degradation, where the Shallow network learns better than the deeper network. Still, in theory, the deeper network can learn more complex features. ResNet addresses this problem by introducing residual blocks and residual connections (Skip connections) to the network. Another problem that ResNet addresses for the shallow network is the early drop of the loss function in the first couple of epochs and a negligible drop in the loss function for the rest of the epoch.

3.1.1 Residual Blocks and Residual Connection

A residual block or ResNet block is a package of internal layers and residual connections. Internal layers include a convolutional layer, batch normalization, and activation function[57]. Convolutional layers perform convolution operations to extract features from the input layer. Based on the network operation and architecture, these convolution kernels can be the size of 3x3, 1x1, or both. To stabilize and accelerate the training, batch normalization is applied after the convolution layer to normalize the activations. Further, to learn complex patterns in the input data, a ReLU(Rectified Linear Unit) is applied after the batch normalization to introduce non-linearity to the model. After passing through the internal layers, the output of the internal layers performs element-wise summation with the actual input through the skip connection. Further, a ReLu is applied to the summation to feed into the next ResNet block. The detailed architecture of the ResNet block is presented in Figure 3.2. The reason behind the use of the skip connection between blocks is that input data arrive at the later layer of the network and make their inputs more meaningful[57]. Also, loss gradient to arrive at the earlier layers of the network and make their updates more meaningful. In other words, the layers will pass their data forward within each block normally, but they will have a new type of connection between blocks. The connection works by combining input to the block and output to the block, giving two different paths of data to follow.

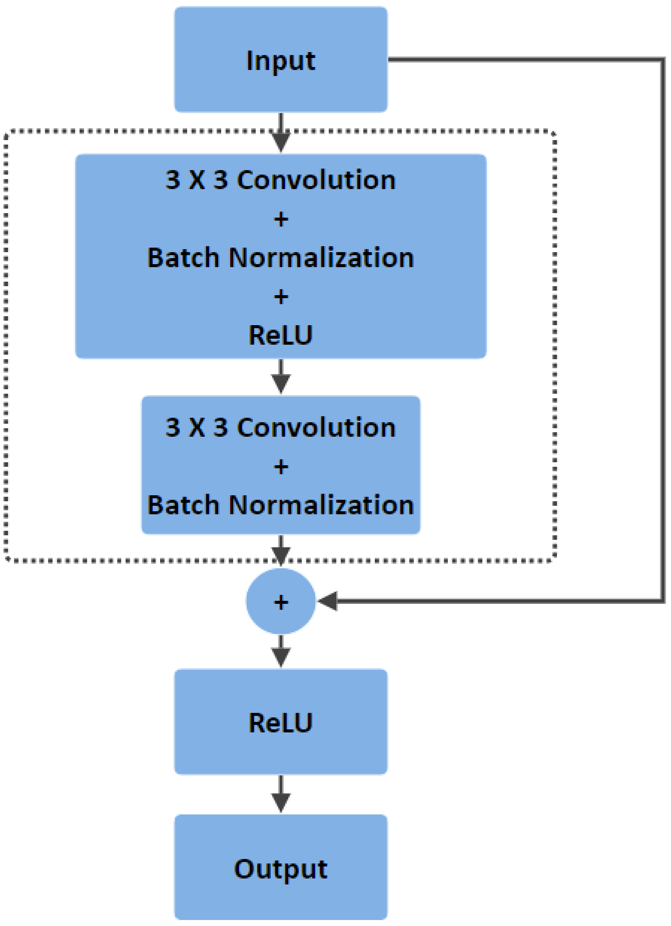


Figure 3.2: Residual block with a residual connection.

To combine the output of the block with the input of the block, a specific method needs to be followed, and it should be a simple function that passes gradients along undisturbed. One solution can be tensor input and tensor output and add them element-wise; the other is to concatenate those tensors. The result is now a network that is built out of a residual block. The skip connection will accelerate training and make the loss graph decay slowly with the number of epochs, not a huge drop in the first couple of epochs.

There is another kind of ResNet block where, instead of using the direct skip connection, the actual input passes through a 1x1 convolution layer. The reason behind using a 1x1 convolution layer in the pathway of skip connection is to solve the shape mismatch of the input and output of the residual block. The details of this ResNet block are presented in Figure 3.3.

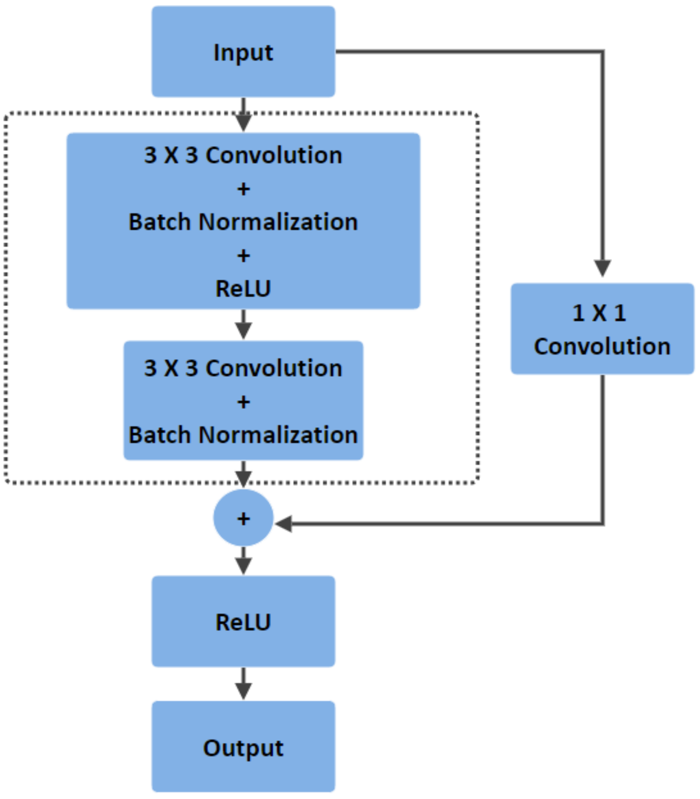


Figure 3.3: Residual block with a residual connection passes through 1x1 convolution layer.

When inputs are added to the output of the first block, due to the convolutional operation in the ResNet block, the channel of the input or the height and width of the input may get changed based on the kernel size, padding, and stride value. This mismatch will lead to an error during the addition or concatenation process. So, some operation is needed to reshape the inputs or only add the inputs to a part of that block's output. It's more complicated when the block contains a convolution layer. It is not wise to constrain the depth of all later layers based on the channels in the input, and it is necessary to make sure that the height and width of the convolution layers are combined with the height and width of the input. So, to use residual blocks, usually, it is desired to have some special case for getting the shape of the input to match up with the shape for the first block, and then it is needed to make sure that subsequent blocks have a common shape so that this sort of problem is minimized [58]. This is why 1x1 convolution is used in the pathway of the skip connection to increase or decrease the number of channels and use appropriate padding and stride to preserve the height and weight of the input.

3.1.2 ResNet Architecture

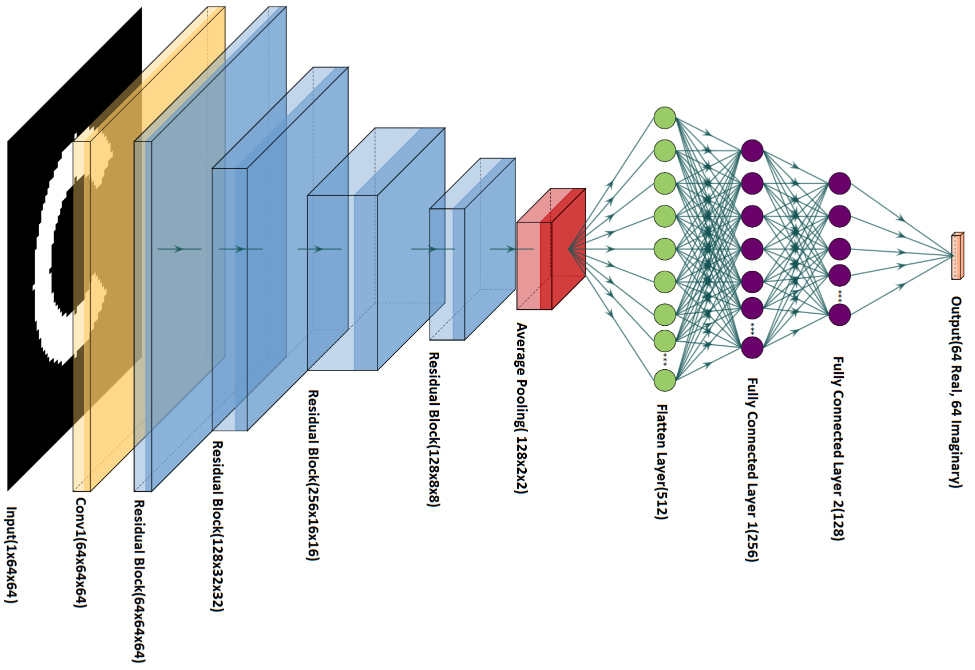
We have considered ResNet-18 in our neural network simulator to train the input data. The input data will be passed through a series of convolution layers before passing through an average pooling layer. It is called ResNet-18 because a total number of 18 layers take part in the operation. Four consecutive residual blocks are placed one after another in the network architecture. Each block consists of convolutional layers, batch normalization, and non-linear activation with the shortcut connection from the input to the block to the output. The network starts with a 7x7 convolution layer with stride two and padding three, followed by 4 residual blocks, before reaching the average pooling layer. The detailed network flow is presented in Table 3.1.

Layer name	**Output size**	**18-layer**
Convolutional Layer 1	64×64	7x7,64, stride=2, padding=3
Residual Block 1	64×64	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
Residual Block 2	32×32	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
Residual Block 3	16×16	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
Residual Block 4	8×8	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
	2×2	Adaptive Average Pool (2,2)

Table 3.1: ResNet-18 architecture with each block's convolutional layers.

It is reasonable to expect that networks built out of residual blocks will be easier to train. The first biggest advantage of a residual block is that the computing function is augmenting the existing data. Since the input is being passed along and will still be available to the later layers in the network, the job of the first block (Residual Block 1) is no longer to figure out everything important about the input that needs to be passed along but rather to figure out what information it can add on top of the input to make subsequent processing easier. This sort of function tends to be much easier to learn because the block doesn't have to start by figuring out what information the input contains; instead, it starts by passing along all of the input and barely modifying it. Suppose concatenation combines the block output with the actual input through the shortcut connection. In that case, the input is passed along unchanged in addition to whatever the block outputs are. Still, even if addition is used instead of concatenation, the layers are initialized with random weights. Those weights tend to be small and centered around zero, meaning that what is added is initially close to zero. The information that is passed initially is relatively unchanged. So, each block has a simpler task to learn, and each block has access to better information from which to learn. Another advantage of using the ResNet architecture is that the gradients have much shorter paths to follow to get to each network layer. Since each block has one path that goes through its layers and one path that goes around them, the gradients will flow along both of those paths, and this means any layer in the network now has a relatively short path by which loss gradients can arrive and usefully update what that layer is updating. Therefore, in the initial epochs of the training, when the information coming along the path that goes through all of the layers is relatively informative. But still, useful updates can be gotten out of the shorter paths and decrease loss immediately. In addition, the later blocks in the network start computing more and more useful functions, and the information coming along this path will get more and more informative. Thus, the gradients continue to decrease and aid in faster training. Lastly, residual blocks have an additional advantage of modularity. It can be seen from Table 3.1 that each of these blocks has essentially the same structure, and all the blocks are short-circuited in the initial training. Adding more blocks and building a deeper, more robust network is relatively easy.

Residual blocks are not the only element in the network architecture, after the average polling layer all the neurons are flattened and then fed into fully connected layers. In the simulator network architecture, two fully connected layers have been used to make the final decision. The complete process flow of the neural network is illustrated in Figure 3.4. The flattened layer's primary objective is to transform the convolution layer's multidimensional output into a 1D vector. This conversion is necessary for the fully connected layers, which take a 1D vector as input, whereas convolution layers take a 2D image or matrix grid as input. In our case, we used a grayscale 64x64 pixel image as input with a single channel. The number of channels increased while passing through the convolution layer and series of residual blocks, and the spatial dimension was reduced gradually. How the spatial dimension of the input image gets changed along with the number of channels is also shown in Figure3.4, mentioned below each of the process blocks, and the comment mentions the output of that particular process block. The spatial dimension remained similar for the first 2 layers because the earliest layers aim to collect low-level features like edges, corners, and textures, and the spatial dimensions are broad. Preserving more significant spatial dimensions makes sense because these attributes are still strongly related to the spatial arrangement of pixels. The network collects more abstract properties as it moves through its layers, such as higher-level patterns and object pieces.



Input(1x64x64)
Conv1(64x64x64)
Residual Block(64x64x64)
Residual Block(128x32x32)
Residual Block(256x16x16)
Residual Block(128x8x8)
Average Pooling(128x2x2)
Flatten Layer(512)
Fully Connected Layer 1(256)
Fully Connected Layer 2(128)
Output(64 Real, 64 Imaginary)

Figure 3.4: Complete diagram of the neural network based on the ResNet-18 architecture

By decreasing the spatial dimensions, the network concentrates more on the characteristics than their precise position. In the meantime, the network can represent a more extensive range of these intricate properties by increasing the number of channels. If the spatial dimensions are not lowered as the network gets deeper, the computational and memory needs will increase quickly. Processing larger feature maps requires more computing power. The network gains efficiency and can support deeper topologies with less resource consumption as the spatial size is reduced. The network has no information loss when the spatial dimensions are reduced. Instead, by adding more channels, the network can record more feature representations at every point in the condensed spatial space. In other words, the network learns richer, more detailed features within smaller regions as the spatial dimensions decrease because of the deeper feature maps. The size of the input tensor that passed through the ResNet-18 is 1x64x64, and the output tensor after the ResNet blocks is 128x2x2. These multidimensional tensors are flattened to 1D tensors to feed to the decision-makers, which is called a fully connected layer. The 1st fully connected layer receives 512 neurons from the flattened layer, and based on the features of those neurons, it passed 256 neurons to the next fully connected layer. Out of those 256 neurons, the final fully connected layer passed 128 neurons as an output where 64 neurons corresponded to the real part and 64 neurons corresponded to the imaginary part of the complex label data.

3.2 Datasets Generation Process

The most crucial part of any deep neural network is to create a dataset by which a deep neural network will be trained. Before that, it's necessary to identify what kind of data is needed to fulfill the network's input and output. As the main goal of our model is to predict the complex electric field that is transmitted through a metasurface, the model should learn the trends, how the metasurface interacts with the incident electric field, and let a portion of the electric field pass through the metasurface. The interaction between the metasurface and the electromagnetic wave is quite unpredictable in the nanoscale and is dependent on the incident electromagnetic wave's wavelength as well as the metasurface's design, shape, size, thickness, and material properties and is calculated by solving Maxwell's equation applying on the entire metasurface. To mimic the behavior of a numerical simulator, we need thousands of metasurfaces and their corresponding simulated results to pass through the neural network, learn the complexity of a numerical simulator, and predict the transmission electric field.

So, the main idea is to design a metasurface, simulate the metasurface using the Ansys Lumerical FDTD simulator, and save the transmission electric field to feed the deep neural network. But here, we are not talking about a single metasurface. We need thousands of metasurfaces with different designs, sizes, and shapes and the corresponding simulated results to forecast the behavior of metasurfaces through deep neural networks. Designing and simulating those metasurfaces one by one is not a practical solution and would take months of labor. We have discovered a way to design a large number of metasurfaces with diversity in their architecture and a way to simulate them together within a short period. The flowchart in Figure 3.5 demonstrates the steps to create a complete single dataset from designing to simulate the metasurface.

We have used some algorithms to automatically generate metasurface unit cells with diverse functionality. The process will be discussed in detail in section 3.2.1. We took a top view of a 2D metasurface and converted the image into a 64x64 pixel grayscale

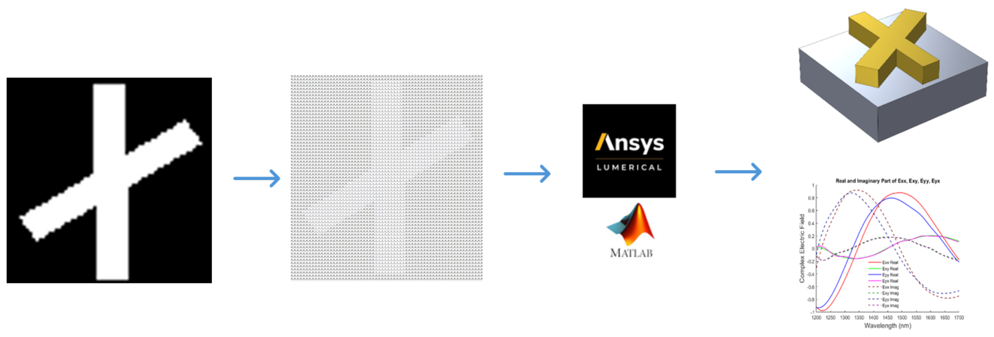


Figure 3.5: A process flow to generate a complete training dataset.

image. We normalized the pixel value to 0 and 1. We mapped the image so that the shape of the metasurface, which is made of Gold (Au), holds 1 in the 64x64 image, and free space holds 0. So, each pixel results in a square of length 5.3 nm on top of the substrate. Based on the pixel mapping, the metasurface is printed on the substrate. We also integrate MATLAB with the Lumerical FDTD to design the metasurface pixel-by-pixel to perform the numerical simulation and save the transmission spectra. We will also discuss the process in detail in section 3.2.2.

3.2.1 Metasurface Generation Method

Since a 64x64 pixel grayscale image will lead to a metasurface, We have introduced various 2D geometry in the datasets that include circles, sectors, complementary sectors, rectangles, triangles, ellipses, hexagons, crosses, rings, split-rings, L shapes and Lunes of different sizes and shapes. We have created an algorithm to deal with these kinds of shapes, where we applied the formula to create those shapes. The algorithm generates those shapes from smaller to larger scales and shifts and rotates the shape within the 64x64 pixel map. The datasets created from the algorithm are presented in Figure 3.6.

We have also introduced another algorithm, which made presenting the 2D geometry with a formula quite difficult. The algorithm takes an RGB image of the geometry and converts it to a 64x64 pixel grayscale image. It extracts the geometric feature from the image and scales the geometry present in the image from smaller to larger size, similar to the previous algorithm it also shifts and rotates the geometry within the 64x64 pixel map. The most significant advantage of this algorithm is that it can create any shape and design of your choice to make it suitable for generating a metasurface. Some of the user-defined geometry created from this algorithm is shown in Figure 3.7.

Lastly, to introduce more variety in the dataset, we have created another algorithm that creates completely random polygons, and the complexity of the polygon is controlled by

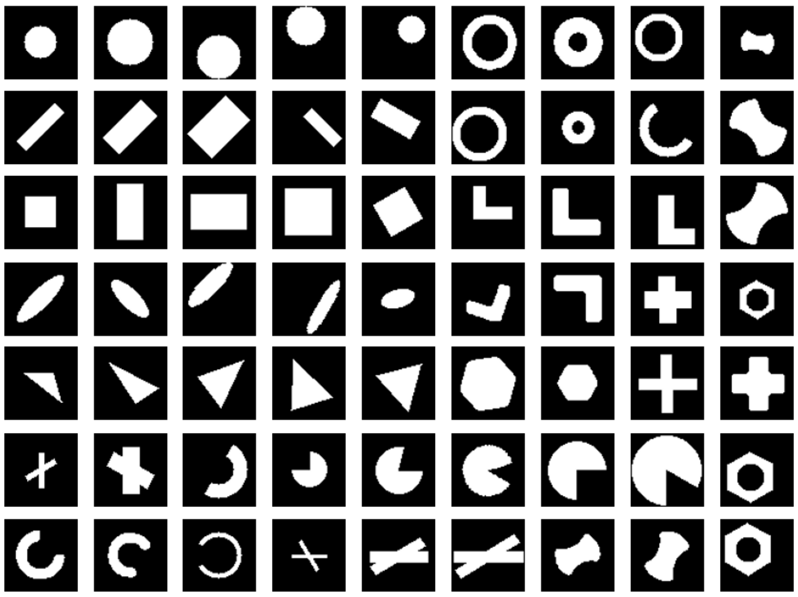


Figure 3.6: Geometries generated from the algorithm using specific formulas .

the number of vertices randomly chosen between 5 and 10. The radius of each vertex from the center is randomized, making each polygon unique. This algorithm provides some unique datasets with random architecture. Some sample from the random generator is shown in row 6 in Figure 3.7.

The main goal behind those algorithms was to introduce thousands of metasurfaces with sharp edges and complex geometry to the neural network. Because sharp edges and complex geometries offer good light-matter interaction and can alter the transmission of the incident light [13]. We wanted our model to learn those complex features during the training to better respond to any arbitrary shape of the metasurface unit cell. Further, we introduced the MNIST datasets[59] to the model; since MNIST datasets consist of numbers from 0 to 9 and all of them are handwritten digits, this creates variance to the datasets, and those numbers have sharp edges and complexity in the shape that is useful for the neural network to learn complex feature. A few samples from the MNIST datasets are shown in Figure 3.8.

Overall, we have generated 8508 datasets for the neural network that we fed to Lumerical FDTD with the help of MATLAB to print the metasurface from the 64x64 pixel map to the Lumerical software, simulate the shape, and save the corresponding results.

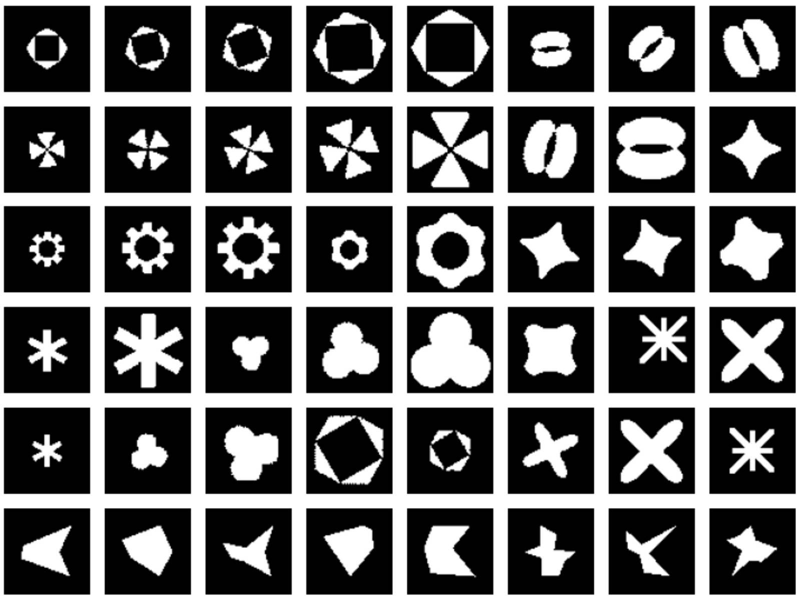


Figure 3.7: Geometries generated from images and random polygon generator.

3.2.2 Metasurface Simulation Method

We have integrated MATLAB with the Lumerical FDTD to simulate the metasurfaces and obtain the transmission spectra numerically. Instead of designing the metasurface manually and simulating it, we wrote a MATLAB script to create all the metasurfaces and simulate them all together one after another. The complete process is shown through a flowchart in Figure 3.9

Firstly, we import the metasurfaces as a 64x64 matrix of zeros and ones in a text file that contains multiple metasurfaces row-wise. We define the substrate dimensions, material properties, metasurface thickness, and material properties globally. The source wavelength span of 1200 nm to 1700 nm was considered a plane wave source. Using a loop each time, the script reads a 64x64 matrix and prints a rectangle where 1 is present in the matrix, and as a result, all the rectangle combines a complete metasurface structure. A boundary condition is applied on the unit cell along with a transmission monitor that records the transmitted complex electric field from 16 points of the wavelength span. Then, the complex electric field is saved in a text file. The script checks if it is the last metasurface and runs all the metasurface 2 times, first with the perfect x-polarized light and then with the perfect y-polarized light. We divided the metasurface batch by batch, containing around 1000 metasurfaces. It took almost 12 hours to simulate a batch with an Intel Core i7 8th generation 32 GB CPU machine. The details of the scripts are provided in Appendix A.

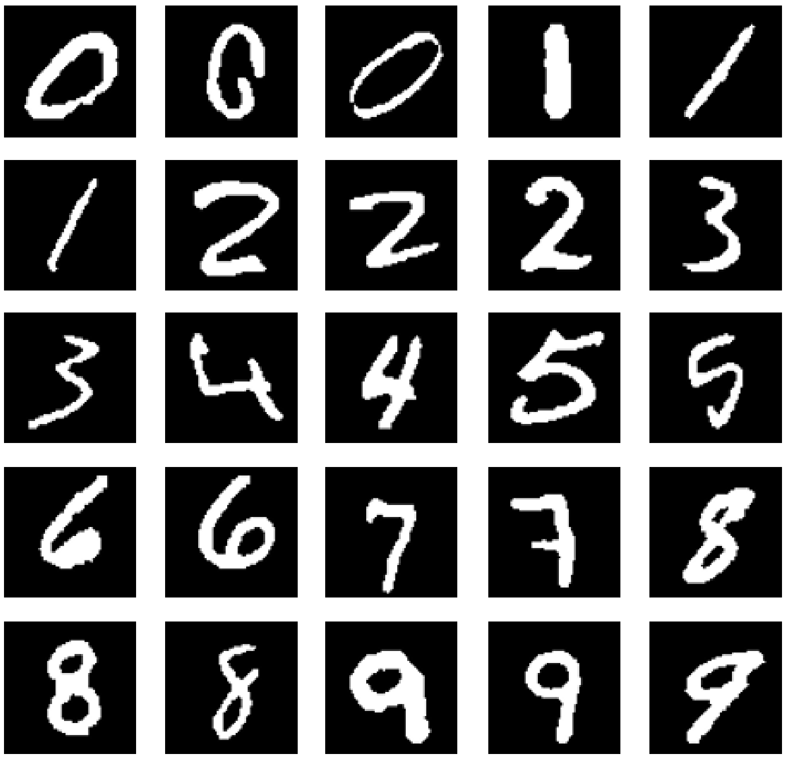


Figure 3.8: Some geometries from the MNIST datasets

3.3 Global Design and FDTD Simulation

The global design includes the substrate dimension and material properties, as well as the metasurface's material properties and thickness. During the design and simulation process, this parameters were kept constant. We have chosen SiO_2 as a substrate material[60]. The length and width of the substrate is 340 nm. The height of the substrate is 300 nm. On the other hand, the material chosen for the metasurface is Gold[60], and the thickness of the metasurface is 50 nm. The parameter list is shown in Table 3.2.

Parameter	L	h	t	Wavelength Span
Value (nm)	340	300	50	1200-1700

Table 3.2: Global Parameter list for metasurface unit cell

A metasurface unit cell with the given description is shown in Figure 3.10. We say these parameters are a global cause except for the metasurface shape and geometry change in every simulation. It was mentioned earlier that we have used Ansys Lumerical FDTD to simulate the metasurfaces numerically. Let's discuss the simulation process in more detail. FDTD is a popular method for simulating the interaction between the electro-

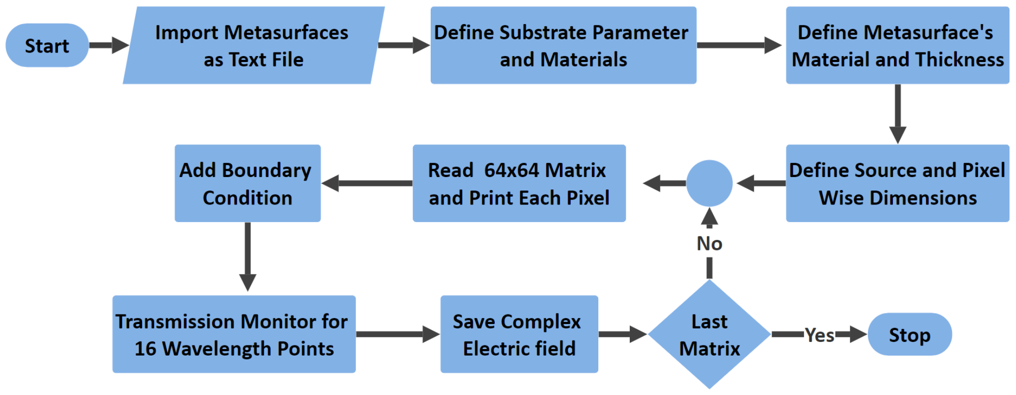

Start
Import Metasurfaces as Text File
Define Substrate Parameter and Materials
Define Metasurface's Material and Thickness
Define Source and Pixel Wise Dimensions
Read 64x64 Matrix and Print Each Pixel
Add Boundary Condition
Transmission Monitor for 16 Wavelength Points
Save Complex Electric field
Last Matrix
No
Yes
Stop

Figure 3.9: Illustration of the process flow to simulate metasurfaces with the help of MATLAB and Ansys Lumerical FDTD

magnetic wave and materials of different topologies. FDTD is used in a wide range of phenomena in optics and photonics, including waveguides, photonic crystals, diffraction gratings, ring resonators, metasurfaces, etc[61]. FDTD operates by solving the Maxwell equation over the simulation region[62]; it describes the dynamics of E-field and H-field on any photonics device. In a simpler manner, when any device is numerically calculated, it solves for the permittivity and electric currents to get the E-field and H-field response of the device. Permittivity describes how the materials are arranged in space and the electric currents (moving charges) that resemble the electromagnetic wave injected into the device. Our metasurface unit cell is a computational domain in Figure 3.10. FDTD will discretize the whole area into thousands of small cubes called Yee cubes[63] and then describe the fields in the discrete lattice form of this cube. The electric field is distributed on the edge of the Yee cube, and the magnetic field is distributed on the surface. The length of the Yee cube depends on the wavelength of the incident electromagnetic wave. Generally, it is considered $\lambda/20$, and the maximum length is considered $\lambda/5$; the smaller the cube, the finer the results, but the smaller cube takes a longer time and requires large computational memory. In Ansys Lumerical, it is considered the mesh size. We have used the mesh accuracy index 2, a moderate mesh accuracy where 10 mesh cells are used in the computational region for minimum wavelength. Since we will simulate thousands of metasurfaces, we have used a moderate index value considering the simulation time.

We have used a plane wave as a source to radiate the computational domain. Preliminary the metasurface unit cell is illuminated with the perfect x-polarized light and further with the perfect y-polarized light. A boundary condition is set around the computational domain so that after the interaction of the plane wave with the computa-

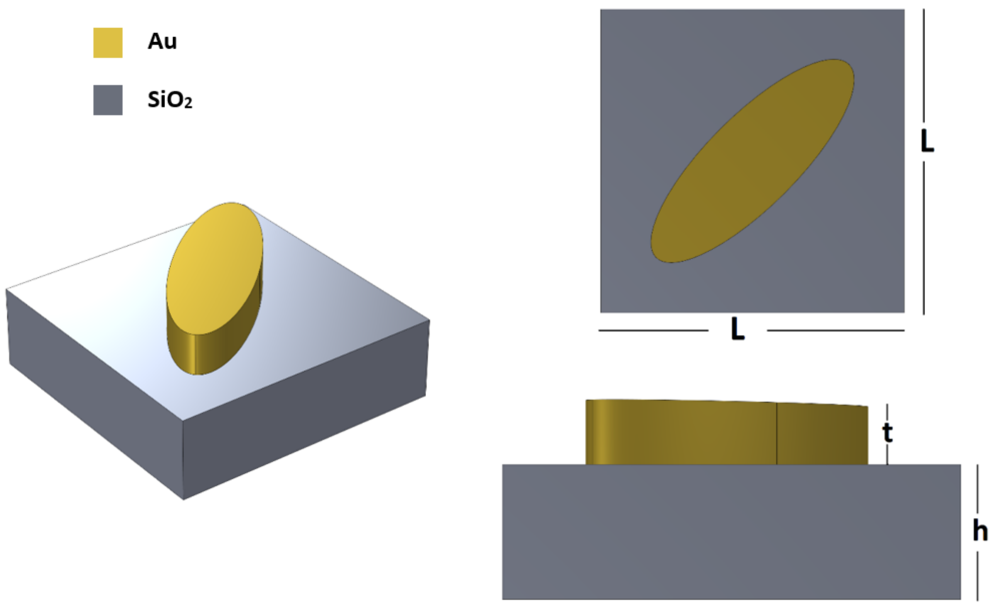


Figure 3.10: Demonstration of a metasurface unit cell

tional domain, no wave can come back and interfere with it again. A Perfect Matched Layer(PML) is applied on both the +z and -z directions of the computational domain. PML is a specially designed absorber that can absorb electromagnetic waves from any angle of incidence. In general, the thickness of this PML is 10 times of the Yee cells. As the metasurface is a periodic array of metasurface unit cells of a similar kind, it is not necessary to simulate all the unit cells, so the x and y are kept periodic in the boundary condition. In short, the plane wave source is kept above the metasurface unit cell. A monitor is placed below the unit cell to record the transmitted electromagnetic wave from the metasurface. The monitor computes the Poynting vector flux that passes through the monitor plane as a function of frequency. To avoid computational difficulties in the neural network, we recorded the transmission spectra for 16 wavelength points within the wavelength range of 1200 nm to 1700 nm.

4 Results and Discussion

To simulate a wide range of metasurfaces in a short period, we have introduced a ResNet-18-based neural network simulator. The neural network takes a 64x64 pixel grayscale image of a metasurface and predicts the complex electric field transmitted. The neural network is trained based on the Gold metasurface to anticipate the transmission spectra within the optical communication wavelength. Thousands of simulated data are used to train the model, which can predict the transmission spectra with considerably high accuracy. The purpose of the neural network is not to take the place of the traditional numerical simulator like Ansys Lumerical FDTD but rather to introduce a time-convenient approach to serve the related research goal. As mentioned before, a Compositional pattern-producing Network (CPPN) will be built in the future with the help of a Generative Adversarial Network (GAN) to produce a metasurface that serves a specific purpose, as an example rotation of the polarization state of the incident light to any arbitrary choice of polarization. This CPPN will generate a compositional image of metasurfaces with the combination of two or three metasurfaces to obtain a specific transmission profile. Since the CPPN will generate hundreds of compositional patterns to simulate those patterns quickly and obtain the transmission spectra, this ResNet-18-based neural network simulator will be used. The detailed training process and performance of the simulator network will be discussed in the later sections.

4.1 Training Process

The training of the neural network was performed on the 7522 datasets. During the training, the batch size of the data to train in each load was 16, and every time, the data was shuffled randomly so that the neural network could not memorize the data and alter the prediction. The training process was performed on 100 epochs. The training took roughly forty minutes and was conducted on an RTX 5000 16 GB GPU computer. The number of epochs counts is how many times the neural network will see a single dataset during the entire training period. Consequently, during the training session, which spans 100 epochs, the model will handle 471 batches of data per epoch and complete 47,100 iterations overall. The neural network is trained with each batch and updates the gradient through backpropagation.

The learning rate used during the training is 0.0001. Learning rate plays a very important role in the training process; after calculating the error between the actual data and predicted data, it updates the gradient by the step size of the learning rate. To obtain a minimum loss, the value of the learning rate is crucial. If the learning rate is high, the model parameters change too fast, and the minimum point can be overshot. The parameters will oscillate around the minimum error or may diverge. Though the training process will be faster, the prediction will not be accurate. The training process will be sluggish if the learning rate is too low. It may become stuck in local minima due to the delayed parameter updates, which prevents the model from reaching the global optimum. We set a learning rate that leads us to not too slow convergence to minimum loss and faster training. The hyperparameters of the model during the training are shown in Table 4.1.

Hyperparameter	**Value**
Number of Epochs	100
Learning Rate	0.0001
Batch Size	16
Optimizer	AdamW
Activation Function	ReLU
Loss Function	MSE Loss

Table 4.1: Hyperparameters used in the training process

A widely used activation function ReLU is used to introduce non-linearity to the model, non-linearity helps the model to learn complex functions to capture the patterns in the datasets. AdamW is used in our model as an optimizer, an improved version of Adam optimizer. In ResNet, the AdamW optimizer improves on the Adam optimizer by separating weight decay from the gradient updates. This allows for enhanced regularization and generalization. Large, complex networks are a good fit for AdamW because of its adjustable learning rates, which stabilize training and speed up convergence. Faster training and better performance are achieved by its decoupled weight decay, which successfully guards against overfitting while guaranteeing effective and stable optimization. The loss is calculated based on the mean-squared error (MSE) loss, as discussed earlier. The network's target was to predict the complex label data. It separately predicts the real and imaginary parts of the complex label data. The MSE loss over the number of epochs for the real and imaginary parts is shown in Figure 4.1 and 4.2 respectively.

Figures 4.1 and 4.2 both demonstrate how the training dataset's and validation dataset's real and imaginary portions of the loss graph have the same appearance. It is evident that since the real and imaginary components are being predicted by the same neural network model at the same time and processed by the same network layers, there may be significant similarities in the learning dynamics between the two. As a result of applying identical weights and procedures to each, the loss for each section should naturally follow a similar trajectory. Additionally, the real and imaginary sections of the model

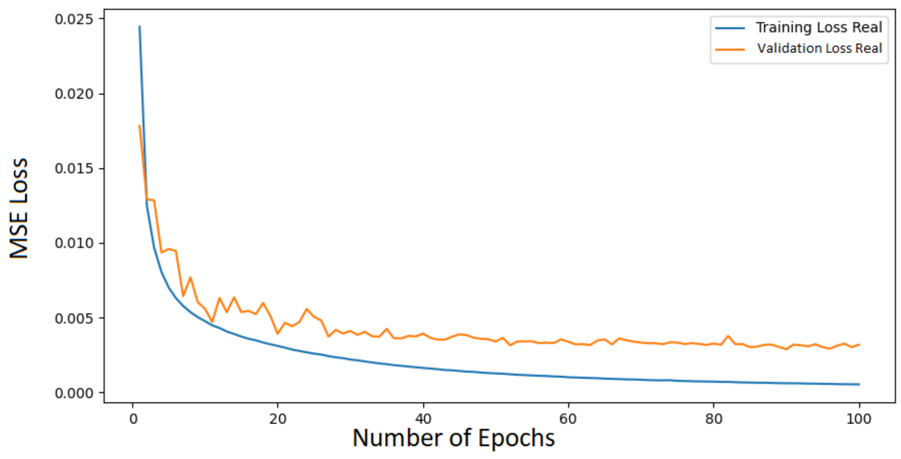


Figure 4.1: Training loss and validation loss over the number of epochs for the real part of the complex electric field

are treated symmetrically using the same MSE loss function, and the optimization process almost perfectly matches how these components were handled. As a result, the loss behavior for the real and imaginary sections was similar throughout epochs.

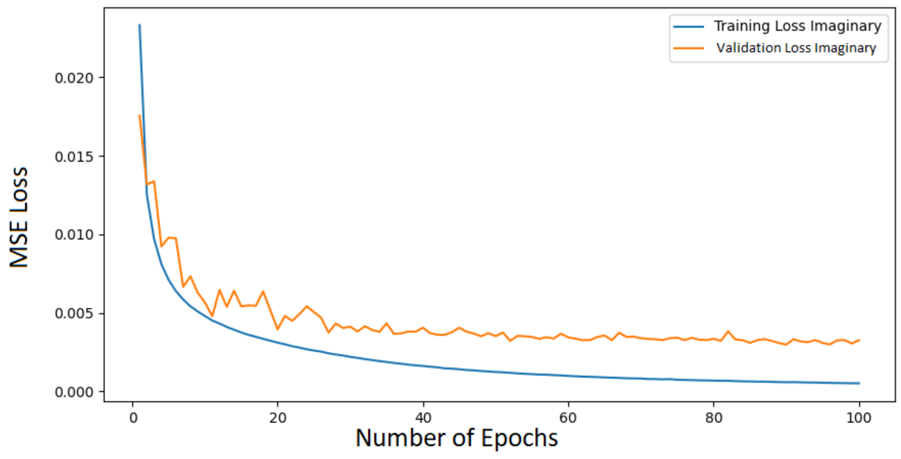


Figure 4.2: Training loss and validation loss over the number of epochs for the imaginary part of the complex electric field

The model is rapidly identifying essential properties from the data in the early stages, as evidenced by the substantial drops in training and test losses, the blue and orange lines in Figure 4.1 and 4.2 indicate the training and validation loss respectively. The model's ability to learn efficiently is primarily attributed to ResNet-18's inventive use of residual connections, which prevent issues with gradients decreasing along the layers. The model appears to have reached a state of generalization to previously unseen data when, after approximately 30 epochs, the validation loss begins to level off and settle. Additionally, the model is becoming more adept at fitting the training data without overfitting, as seen by the steady decline in training loss. A significantly positive development is the narrow difference between test and training losses. This indicates that ResNet-18's design effectively balances regularization and complexity, allowing the model to manage real and imaginary parts without experiencing overfitting or noise. As a whole, this graph indicates that ResNet-18 is a reliable option for complex label data prediction.

4.2 Performance on Validation Datasets

After completion of the training, the performance of the model is validated with the validation datasets. It consists of 836 datasets that didn't take part in the training session of the model and were chosen randomly from the data loader. The scatter plots in Fig. 4.3 and 4.4 display the relationship between the actual data and predicted for the real and imaginary part of the complex electric field, respectively. Each figure consists of four subplots where the subplot shows the prediction statistics for the Eij; here, i represents the polarization state of incident light, and j represents the polarization state of transmitted light. We have presented the prediction statistics for Exx, Exy, Eyy, and Eyx separately for the complex electric field's real and imaginary parts. In the scatter plot, the red dots represent the actual label data obtained by the FDTD simulation, and the green dots represent the corresponding predicted data obtained by the neural network simulator.

For the model to be completely accurate, all green dots should ideally fall exactly on the red dots along the diagonal of the subplots. However, accurate forecasting is unattainable in real-world situations. Most of the green dots in the subplots for both the real and imaginary parts closely trail the red dots, indicating that the model's predictions are generally accurate and fit the data well. If we look closely at the subplot a) and c) for both Figure 4.3 and 4.4, for Exx and Eyy, most of the green dots remain very close to the red dots. This indicates that the model's prediction is quite good if the incidence and transmitted light have a similar polarization state. Although there is some dispersion, it is not too problematic because the model identifies the key trends in this data. On the other hand, for Exy and Eyx in the subplot b) and d) for both the real and the imaginary parts, it seems the predicted green dots are more scattered around the red dots in the diagonal line. The predictions are still largely accurate since the points mostly follow the red line's trend. Still, the spread shows that the model performs a little worse when the transmitted and incident light have polarization states that are rotated 90 degrees.

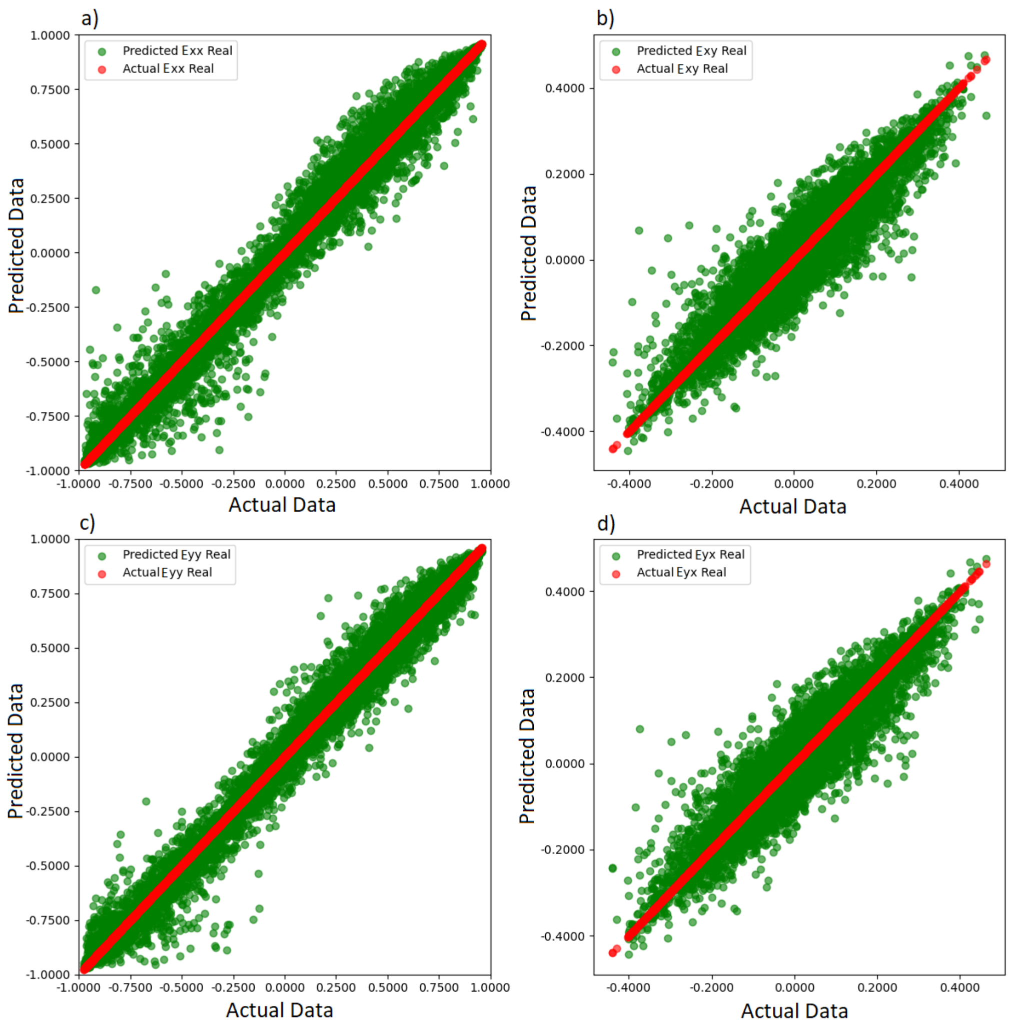


Figure 4.3: Actual vs predicted statistics on validation datasets for the real part of the complex electric field. a) incident as x polarized and transmitted as x polarized b) incident as x polarized and transmitted as y polarized c) incident as y polarized and transmitted as y polarized d) incident as y polarized and transmitted as x polarized

The Exy and Eyx components of the data may have higher variability, or the model is less sensitive to particular features; it might be more difficult for the model to capture the complexity of the data in these areas.

After close observation of the actual and predicted Exy and Eyx data for both real and imaginary parts, we have found that a large number of metasurfaces have the value of the complex electric field for both real and imaginary parts between 0 and 0.1 and their structures are more symmetric that means most of those metasurface can't rotate the polarization state of the incidence light by 90 degrees. The geometry of the nanostructures in a metasurface intended to interact with x-polarized light is impor-

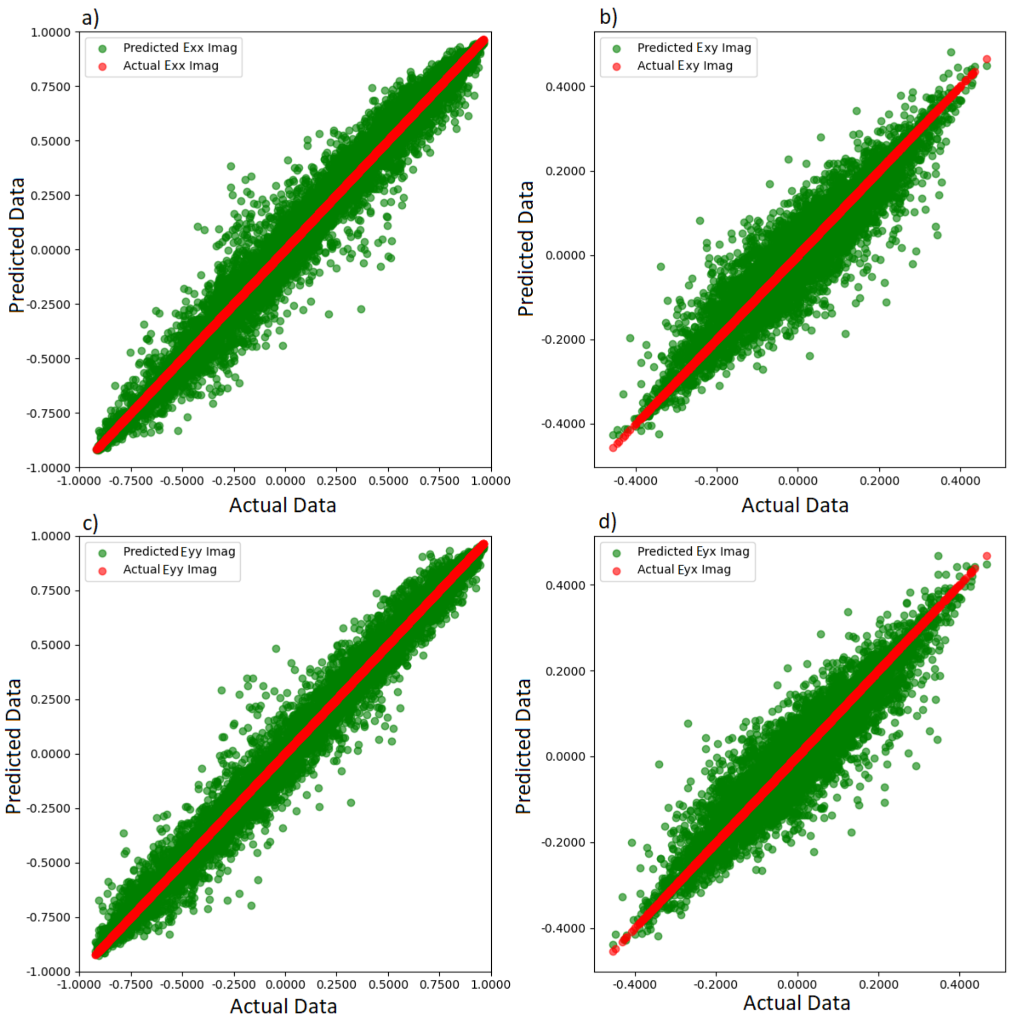


Figure 4.4: Actual vs predicted statistics on validation datasets for the imaginary part of the complex electric field. a) incident as x polarized and transmitted as x polarized b) incident as x polarized and transmitted as y polarized c) incident as y polarized and transmitted as y polarized d) incident as y polarized and transmitted as x polarized

tant in determining whether the polarization of the transmitted light will rotate by 90 degrees, changing from x-polarized to y-polarized. Because they disrupt the symmetry between the x and y axes, asymmetric designs like split-ring resonators, elliptical nanoantennas, and L-shaped structures can produce this polarization shift. By creating an asymmetric reaction to the electric field of the incident light, these forms successfully couple the x-polarized light into a y-polarized state. Conversely, symmetric shapes, such as squares, rectangles, or circles, keep the original polarization since they interact with light in all directions equally. As a result, they do not change the polarization state. Thus, a 90-degree rotation in polarization can only be achieved by structures that introduce asymmetry; very symmetric designs cannot cause this kind of alteration. But where is the issue if the metasurface doesn't alter the polarization state? As mentioned earlier, a considerable number of the real and imaginary parts of Exy and Eyx stay between 0 and 0.1. We range the data to four decimal points that lead us to 0.0000 to 0.1000. For instance, in actual data like 0.0001 or 0.0010 the prediction is bad because when dealing with very small numbers, even slight errors in the model's prediction can appear significant relative to the magnitude of the value. For instance, a small deviation from 0.0010 to 0.0015 represents a 50% error, which is proportionally large. Because of this, the model cannot predict smaller values with the same level of accuracy as it can larger ones, for which the relative error would be substantially lower. This issue might be resolved if the data collection had two decimal point values.

The general pattern is nevertheless the same in all eight subplots of Figure 4.3 and 4.4. Though some fine-tuning may be necessary to address the more complex features of the Exy and Eyx components, the model generally predicts each component well. When generating somewhat correct predictions for most of the data, the ResNet-18 model seems to be learning effectively overall.

4.3 Performance on Test Datasets

We have created a total of 8508 datasets. Before beginning the training process, we removed 150 datasets from the main datasets to create a test dataset; in these 150 test datasets, we tried to keep all types of geometries, at least one from each kind. The model has never seen these test datasets during training or validation. The purpose behind creating this dataset is to do a final evaluation of the model's prediction accuracy. The true value and predicted value on the test datasets are shown in Figure 4.5 and 4.6 using scatter plots where Figure 4.5 represents the real part and Figure 4.6 represents the imaginary part of the complex label data. Similar to the validation datasets, the red dots represent the true value obtained by FDTD simulation, and the green dots represent the predicted value obtained by the neural network simulator. In both figures, subplots a,b,c, and d represent Exx, Exy, Eyy, and Eyx, respectively.

Looking at these scatter plots, which compare the actual and predicted values for 150 test datasets, the model is performing quite well. The green dots, representing the model's predictions, closely follow the red dots, representing the actual data. The predictions for the Exx and Eyy label data are particularly accurate, with most points clustering tightly around the red dots, indicating that the model is capturing these parts of the data very well. On the other hand, for the Exy and Eyx label data, there's a bit more scatter in the predictions. While the model still follows the overall trend, it struggles

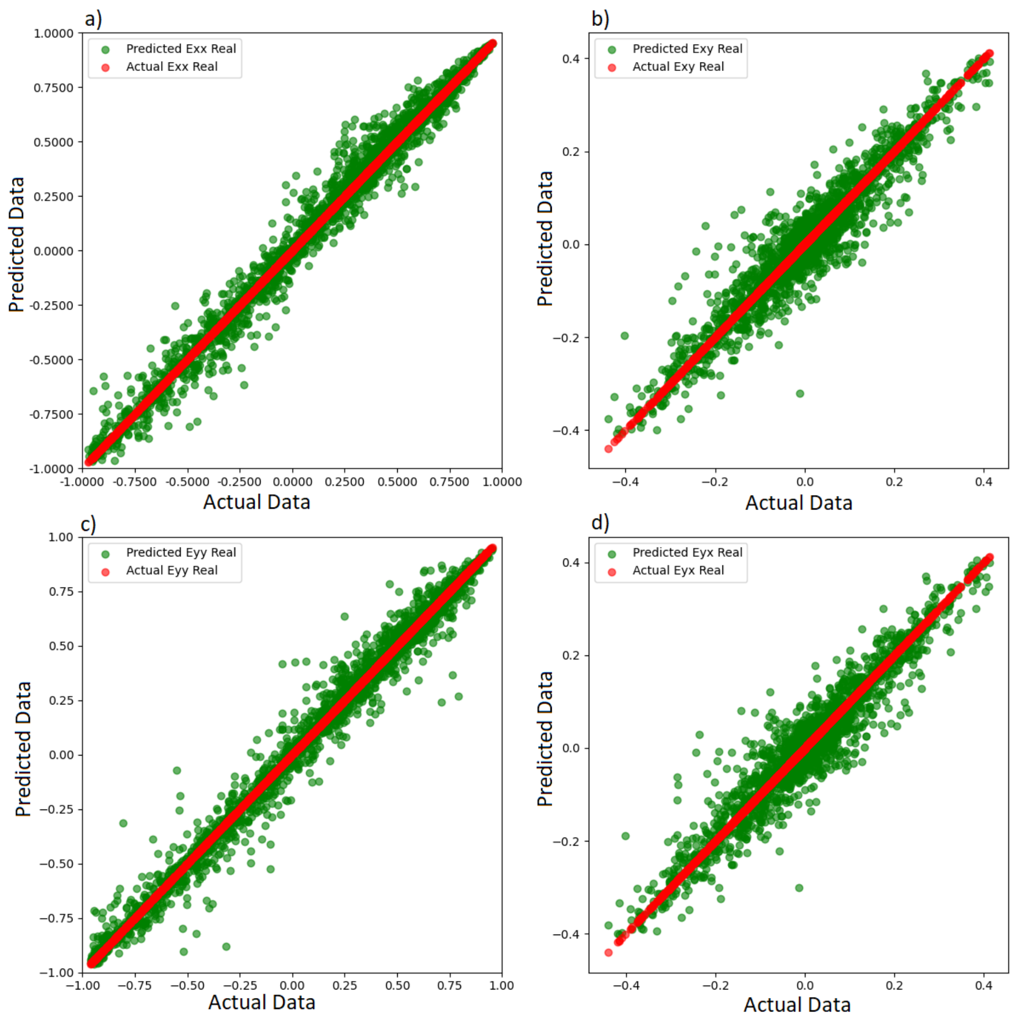


Figure 4.5: Actual vs predicted statistics on test datasets for the real part of the complex electric field. a) incident as x polarized and transmitted as x polarized b) incident as x polarized and transmitted as y polarized c) incident as y polarized and transmitted as y polarized d) incident as y polarized and transmitted as x polarized.

slightly more with these components.

However, the fact that the model is still fairly accurate for these trickier parts is promising. Overall, the model seems to generalize well to new, unseen data, maintaining the solid performance we observed in the validation stage, especially for the Exx and Eyy label data, while handling the more challenging Exy and Eyx reasonably well. This demonstrates that the model has effectively transferred from the training and validation stages to the testing stage.

4.4 Evaluation of The Model

An accuracy test was performed on the test datasets. To avoid the complexity in the calculation, we performed the accuracy test on the absolute value of the complex elec-

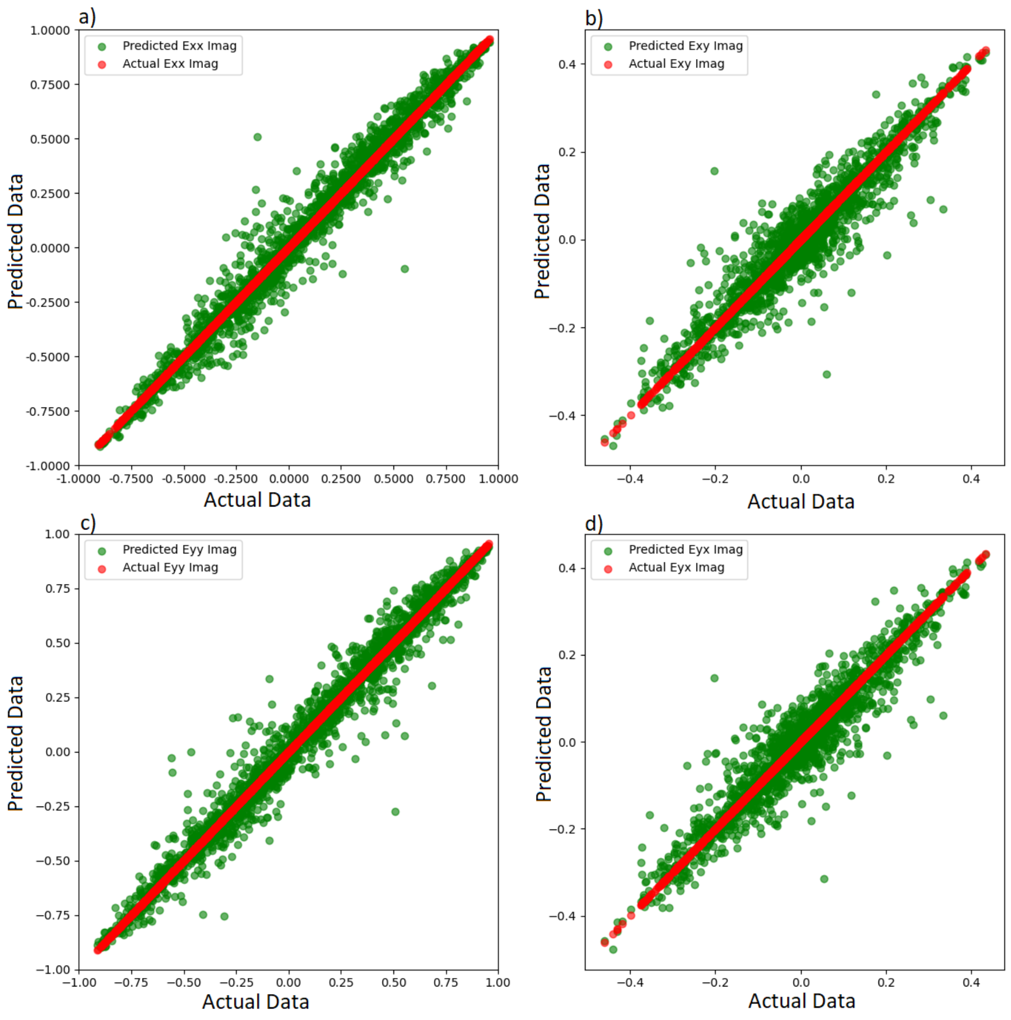


Figure 4.6: Actual vs predicted statistics on test datasets for the imaginary part of the complex electric field. a) incident as x polarized and transmitted as x polarized b) incident as x polarized and transmitted as y polarized c) incident as y polarized and transmitted as y polarized d) incident as y polarized and transmitted as x polarized

tric field. The model's average prediction accuracy is 85.27%, and the average distance between the actual and predicted data is 0.0359. The maximum accuracy was found to be 97.56%, and the minimum accuracy was found to be 44.60%. The average accuracy could have been high if we had set the label data to two decimal points; the reason behind this is mentioned in the earlier section. To check the performance of the model on individual datasets, we plot the actual and predicted values against the wavelength span. This approach shows how the model performs as a neural network simulator on a single metasurface unit cell. In all the figures presented below from Figure 4.7 to Figure 4.15, the solid lines represent the actual value simulated by the FDTD method, and the dotted lines represent the predicted value by the neural network simulator.

4.4.1 Performance on Regular Datasets

Regular datasets are the standard geometries that are essentially employed as metasurface unit cells for a variety of research purposes in several research articles [64, 65, 66, 67]. The datasets comprise hundreds of these geometries to make the neural network simulator aware of the situation and what types of geometries it will deal with.

In Fig. 4.7, we have compared the actual and predicted transmission spectra for a nano ellipse metasurface unit cell. Instead of showing the complex electric field, we have presented the amplitude and phase of the transmitted electric field. Subplot 4.7(a) represents the magnitude, and subplot 4.7(b) represents the phase. We can observe how closely the model's predictions match the actual transmission coefficients in the first plot on the left. The black circles display the projected Exx values, while the red line represents the actual Exx values. The graph shows that the model follows the real curve rather well despite occasional minor variations, especially after the initial drop. The blue circles show the expected Exy values, whereas the green line shows the actual Exy values. Once more, the model performs pretty well, but there is some variation in how closely it resembles the real curve, particularly at the shorter wavelengths.

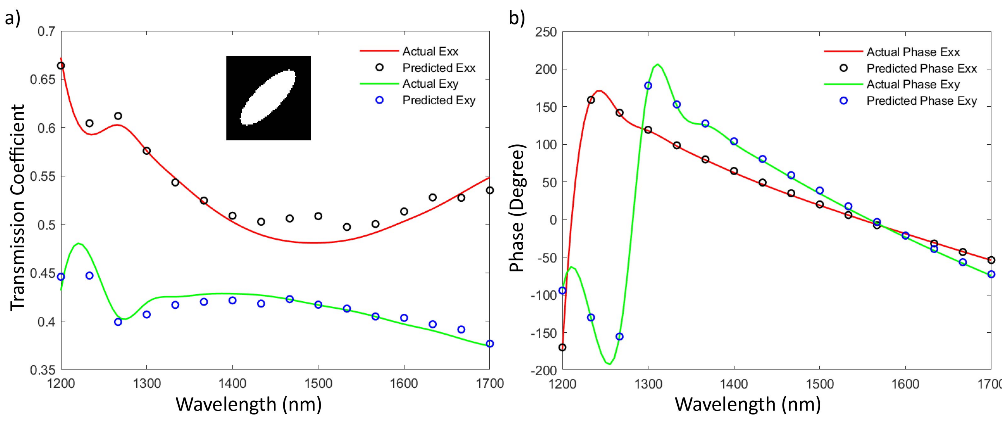


Figure 4.7: Transmission spectra of elliptical metasurface. a) Actual and predicted amplitude of Exx and Exy b) Actual and predicted phase for Exx and Exy.

The phrase comes into focus in the second plot on the right 4.7(b). In this case, the red line shows Exx's actual phase, while the black circles show its anticipated phase. Across the wavelength range, the predictions match the real data rather well, including catching the notable phase shifts. The blue circles indicate the expected phase for Exy, whereas the green line shows the actual phase. The model has several discrepancies, especially at abrupt phase transitions, even though it appears to represent the overall trend of the phase shifts for Exy.

In Fig. 4.8, we have compared the L-shaped metasurface unit cell's actual and predicted transmission spectra. Plotting the transmission coefficient begins with the graph on the left 4.8(a). The black circles represent the model's predictions, while the red line tracks the actual values of Exx. Although there are some small variations, especially in the areas where the curve bends, we can see that the predictions generally match the actual data. The blue circles represent the model's predictions, while the green line shows the actual values of Exy. In this case, the projected values flatten down compared to the actual curve in the middle wavelength region, where the model has greater difficulty capturing the variances. Though there is a difference in the actual and predicted values, the predicted values followed the trends well.

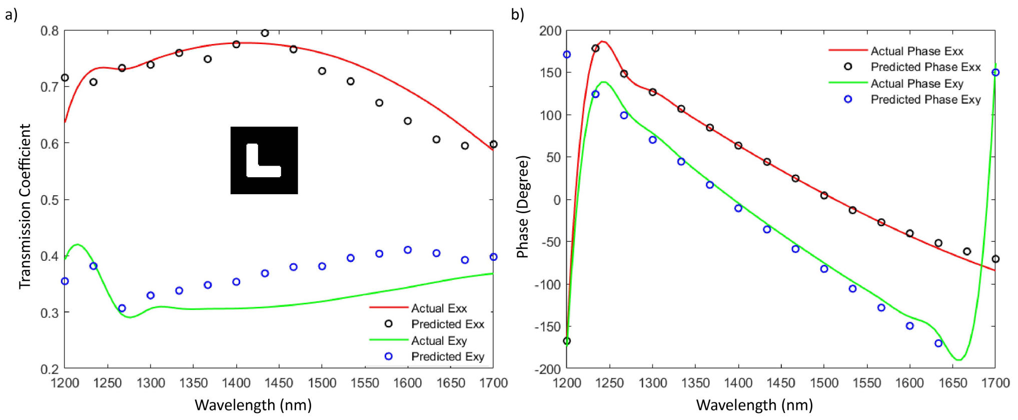


Figure 4.8: Transmission spectra of L shape metasurface. a) Actual and predicted amplitude of Exx and Exy b) Actual and predicted phase for Exx and Exy.

Moving on to the plot 4.8(b), which depicts the phase, for the phase Exx, the actual and predicted values matched well. A slight inaccuracy appears at the beginning of the phase of Exy; the phase transition doesn't fit well, but the rest of the prediction shows a good approximation of the actual data.

Fig. 4.9 presents a circular split ring metasurface. It can be seen from the subplot 4.9(a) that for both Exx and Exy, the predicted value fits nicely with the actual data. But for the Exx, the predicted value is slightly red-shifted, still following the actual trend.

Examining the phase response across the wavelength range, plot 4.9(b) shows some interesting behavior. Exx and Exy exhibit strong phase shifts, with the largest fluctuation occurring at lower wavelengths (about 1250 nm). The phase shifts get smoother and more gradual as the wavelength grows. The overall trend in both cases follows a diminishing phase shift with increasing wavelength, with a few notable peaks and troughs, despite variations between the actual and projected phases.

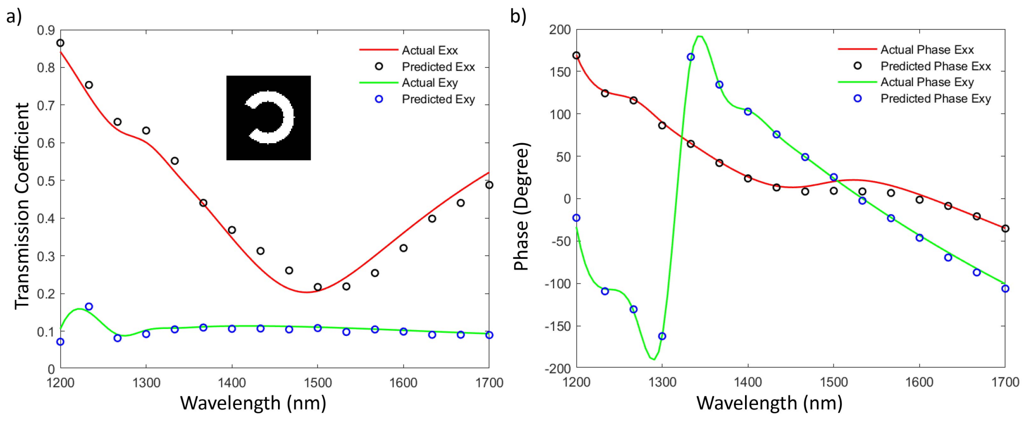


Figure 4.9: Transmission spectra of circular split ring metasurface. a) Actual and predicted amplitude of Exx and Exy b) Actual and predicted phase for Exx and Exy.

We have presented three random regular geometries out of hundreds. By analyzing the three graphs from the regular shapes, it is clear that the model predicts these usual geometries well. Though there is potential for improvement in accuracy, particularly at some important locations when bigger deviations are detected, the model generally captures both the amplitude and phase properties well.

4.4.2 Performance on Irregular Datasets

Geometries that are uncommon and not commonly utilized as metasurface unit cells are called irregular datasets. Besides the regular geometries, we have also introduced hundreds of unusual geometries. The neural network should be trained for every possible combination of geometries that a person could think of, as it will serve as a simulator.

Figure 4.10 shows data for an elliptical-dimer form, both actual and anticipated. It can be seen from the subplot 4.10(a) that the actual Exx and predicted Exx look similar. Though it seems like around 1225 nm is a bit under-predicted and around 1550 nm is a bit over-predicted, the difference between the actual and predicted data is acceptable. On the other hand, the model predicts through the entire wavelength span for Exy, but the promising fact is that the prediction follows the trend of actual data.

The phase graph illustrated in subplot 4.10(b) shows that the model handled the phase transition very well, around 1200 to 1300 nm region. For both Exx and Exy, the predicted phase value maintains the actual phase value all the time, even if it follows sudden changes in the actual graph. This shows that the model has caught the underlying physics regulating the phase shifts well, as it matches the actual data quite well in its predictions of these trends.

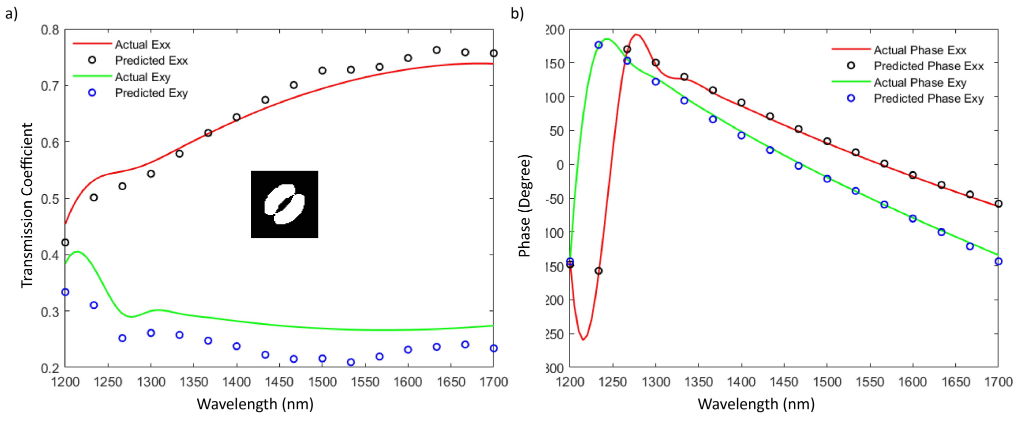


Figure 4.10: Transmission spectra of elliptical-dimer metasurface. a) Actual and predicted amplitude of Exx and Exy b) Actual and predicted phase for Exx and Exy.

In Fig. 4.11, a trimer structure is presented, which consists of three interconnected or closely arranged circular shapes. From subplot 4.11(a), it can be said that, in the beginning, the prediction for Exx was a bit under-predicted, but it fits well gradually as the wavelength increases. For the Exy, the actual value usually stays between 0 and 0.1; even though we have encountered an issue with low-value prediction, the prediction accuracy and graph's trend perfectly match the actual data.

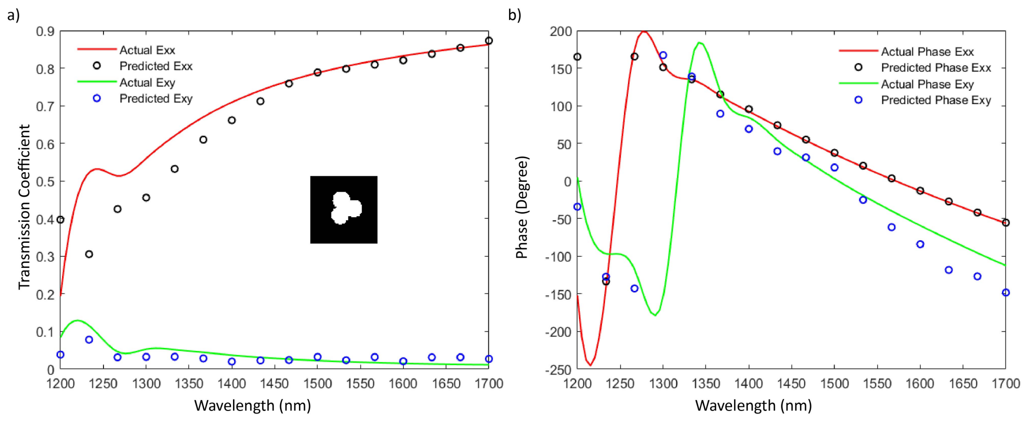


Figure 4.11: Transmission spectra of three closely joint circular metasurface. a) Actual and predicted amplitude of Exx and Exy b) Actual and predicted phase for Exx and Exy.

In the subplot 4.11(b), the actual phase and predicted phase match perfectly for Exx, though there is around a 360-degree phase shift at 1200 nm between the actual and predicted phase. For Exy, predicted values stay around the actual value, and the predicted phase experiences a blue shift compared to the actual data.

We have presented actual and predicted data for a random polygon from the random polygon generator in Figure 4.12. The purpose is to check how the model works on this completely random shape that the model has never seen during training. From the subplot 4.12(a), it is seen that the Exx is more closely under-predicted than the actual data throughout the entire wavelength span, but the graph trends match perfectly with the actual data. A red shift in the predicted data is seen compared to the actual data. On the other hand, the predicted value of Exy matches quite well with the actual data.

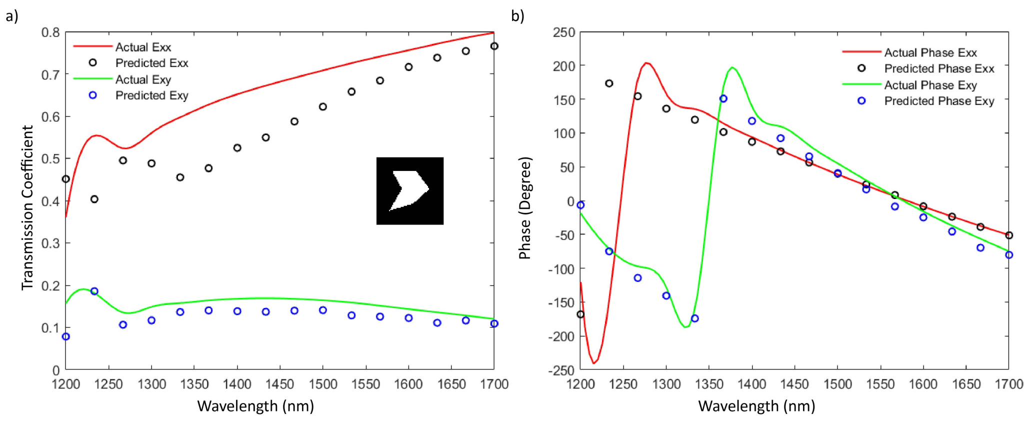


Figure 4.12: Transmission spectra of random polygon metasurface a) Actual and predicted amplitude of Exx and Exy b) Actual and predicted phase for Exx and Exy.

For the phase in subplot 4.12(b), the phase of predicted Exx follows the actual Exx well, but the predicted phase is left-shifted compared to the actual phase. For the Exy, the predicted data perfectly syncs with the actual data and the phase transition around 1350 nm. We have presented three unusual geometries out of hundreds, including a random polygon. The model can predict the transmission spectra with quite good accuracy. Though there are some slight under- or over-predictions, the model follows the overall trends in the actual data really well.

4.4.3 Performance on MNIST Datasets

We have presented some data from the MNIST datasets. MNIST datasets consist of handwritten digits from zero to nine. Those digits are randomly arranged and have sharp edges, sharp turns, and sudden changes in geometry. Predicting the transmission spectra for the MNIST datasets is the real challenge for the neural network simulator. In Figure 4.13, we have presented the actual and predicted transmission spectra for the digit zero. The subplot 4.13(a) shows that the actual data and predicted data merge nicely with each other with a minimal deviation for both Exx and Exy. Also, in the subplot 4.13(b), the phase prediction for both Exx and Exy syncs perfectly with the actual data. Though there is a little shift towards the right in the phase prediction for the Exy the overall prediction accuracy is quite well.

Figure 4.14 shows the actual and expected transmission spectra for digit two. Digit two

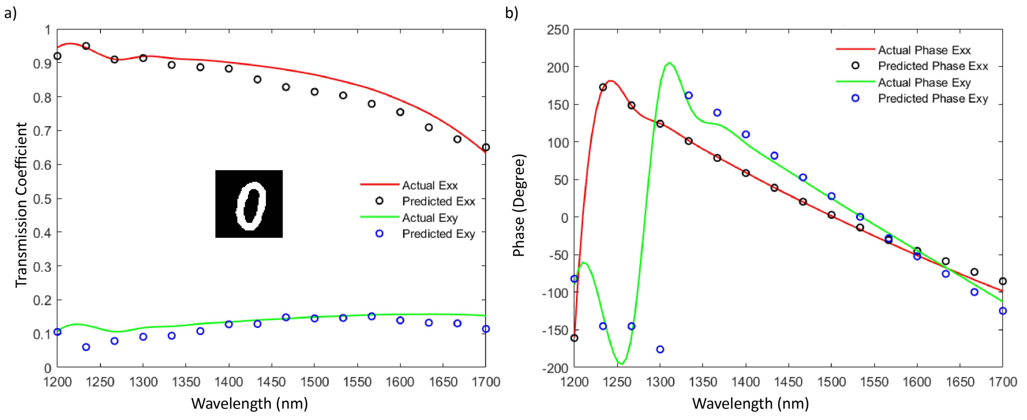


Figure 4.13: Transmission spectra of MNIST digit zero metasurface. a) Actual and predicted amplitude of Exx and Exy b) Actual and predicted phase for Exx and Exy.

has a sharp bend and sharp edge in its pattern, which is a challenge for the model in predicting the transmission spectra accurately. In the subplot 4.14(a), the actual and the predicted values sync well, even for the sudden changes in the actual data for both Exx and Exy. The predicted values follow the trend of the true value quite nicely. It is also seen from the subplot 4.14(b) that the phase of the predicted data for both Exx and Exy matches perfectly with the actual phase. The predicted phase maintains the phase transition nicely in the actual Exy phase, around 1250 nm. Overall, the model shows exemplary prediction accuracy on the digit two, randomly picked from the MNIST datasets.

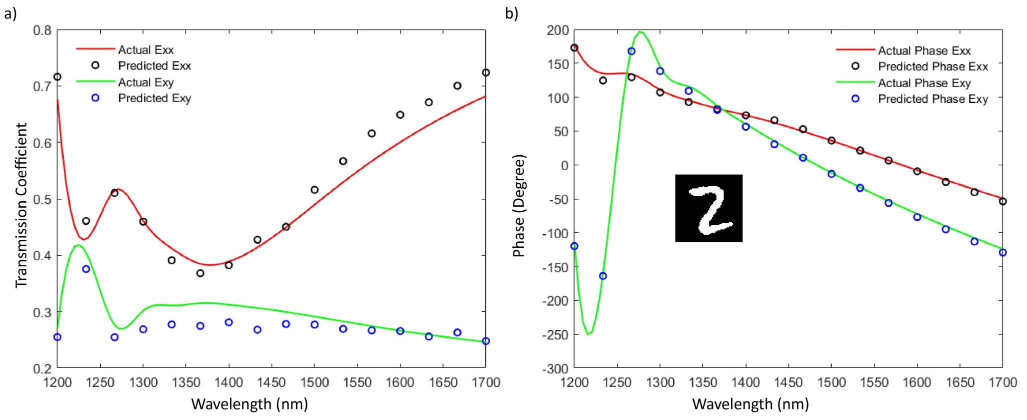


Figure 4.14: Transmission spectra of MNIST digit two metasurface. a) Actual and predicted amplitude of Exx and Exy b) Actual and predicted phase for Exx and Exy.

Lastly, we have presented the actual and predicted data for the digit six in Figure 4.15. From the subplot 4.15(a), the prediction for Exx is quite good, for some points it predicts, and for some points it predicts but overall the prediction graph follows the trend of the actual graph even when the actual value drop suddenly and increases again. Also, for the Exy, the predicted value always maintains the trend of true value with high accuracy and minimal deviation.

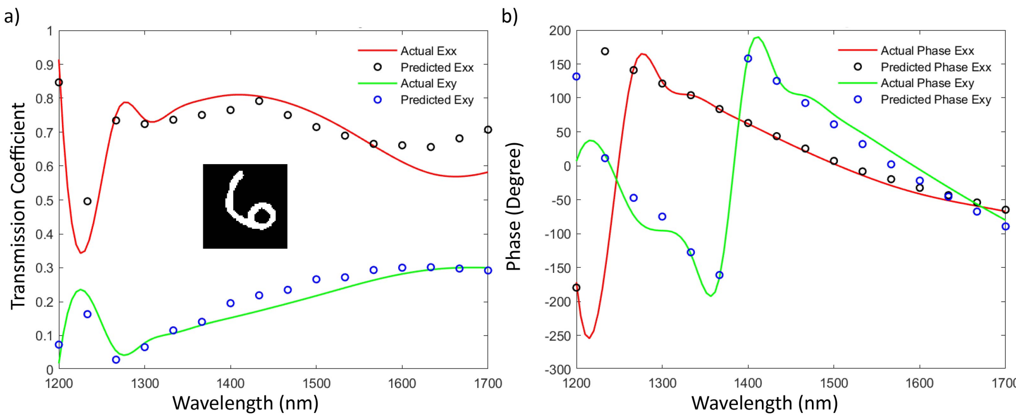


Figure 4.15: Transmission spectra of MNIST digit six metasurface. a) Actual and predicted amplitude of Exx and Exy b) Actual and predicted phase for Exx and Exy.

In the subplot 4.15(b), The predicted phase of Exx maintains the actual phase well. Though the predicted phase shifted slightly left, it maintains all the ups and downs in the actual curve. Also, for the Exy the model handled the prediction nicely, even with the sudden change in the actual data the predicted data syncs perfectly. The predicted data also appropriately maintained the phase transition in the actual data around 1450 nm. Overall, the model's prediction over the MNIST datasets was quite good. The model has learned the complex features from the MNIST datasets and can make accurate predictions. The model is now sensitive to sharp edges, sharp bends, and sudden changes in the pattern, which will help extract features from unseen complex geometry.

We have presented nine samples out of 150 test datasets to show how the model works on different types of unseen data. We have shown the graph concerning Exx and Exy. We have also plotted the data concerning Eyy and Eyx, since the graph looks almost similar in y polarized illumination to avoid making the report bulky we consider not including those graphs in the main report. Some of the graphs concerning y-polarized incident light will be in Appendix B, along with other related graphs. In brief, the model performs well on the test datasets. Though it under or over-predicted and even shifted a bit in some cases, the model predicted accurately most of the time with minimal deviation between the true and predicted values.

4.5 Comparative Analysis

We trained the model with five optimizers to get the optimum response from the neural network simulator. We have compared the performance of several optimizers in terms of training and validation loss, both for real and imaginary parts of the complex label data. The comparative analysis highlights the performance of the optimizer AdamW in contrast to others. The training loss and validation loss based on the optimizer are presented in Table 4.2. AdamW is the most effective at minimizing error during the training phase since it exhibits the lowest training loss in both real and imaginary parts. Additionally, it has one of the lowest validation losses, indicating that the model performs well when applied to fresh, untested data. Because of this, AdamW is a reliable option for estimating metasurface transmission spectra. RMSprop struggles more during validation than during training when it performs rather well with a relatively low training loss. Since RMSprop's validation loss is larger than AdamW's, it may overfit the data or have difficulty generalizing as well.

Optimizer	**Training Loss**		**Validation Loss**	
	Real	**Imaginary**	**Real**	**Imaginary**
AdamW	0.0005	0.0005	0.0032	0.0032
RMSprop	0.0008	0.0008	0.0120	0.0120
Adam	0.0007	0.0007	0.0039	0.0039
SDG	0.0243	0.0237	0.0243	0.0243
Adagrad	0.0109	0.0109	0.0114	0.0114

Table 4.2: Training and validation loss for different optimizers

With marginally more training and validation losses, Adam performs similarly to AdamW. Although it is still a formidable opponent, AdamW outperforms it in both the validation and training stages. SDG shows the largest loss values during the training and validation stages. Although this optimizer might be useful in some situations, it doesn't seem as well-suited for forecasting metasurface transmission spectra because of slower convergence or challenges with efficiently lowering error. While Adagrad outperforms SDG, it is still much less efficient than AdamW, Adam, or RMSprop. Its test loss is comparatively higher than that of some other optimizers, suggesting it is not as generalizable. The prediction statistics like Figure 4.3, will be given in Appendix B for the comparison between the optimizers in terms of accuracy statistics.

Based on the information presented in Table 4.2, AdamW is the most efficient optimizer for the ResNet-18 neural network when making predictions for the transmission spectra of metasurfaces. Concerning training and generalization, it performs exceptionally well, yielding the lowest training loss and among the lowest validation losses. This supports the ultimate decision to choose AdamW as the model's optimizer.

5 Conclusion

In brief, we have introduced a ResNet-18-based neural network simulator. The neural network mimics the conventional numerical simulator for a definite design objective. The neural network predicts the transmission spectra of the Gold metasurface with high accuracy within the 1200 nm to 1700 nm region in a timescale of milliseconds. We have introduced thousands of simulated datasets with varied geometries to train the neural network to predict with high accuracy. After an iterative training process, the model successfully passed the validation and test phase. The training process includes random polygons, the MNIST datasets, and the regular geometries to introduce the model with sharp edges, bends, and complex structures. The model can accurately predict the geometries that it has never seen before. The training was performed in an RTX 5000 16 GB GPU machine, and it took about 40 minutes to train the model with around 8000 datasets. After the training, the model took 0.405 seconds to predict transmission spectra for 150 test datasets. The average accuracy on the test data was found 85.27%. The presented neural network outperforms the traditional solver by approximately five orders of magnitude regarding simulation time. Overall, the neural network simulator performs considerably well in predicting the transmission spectra of unseen complex random metasurfaces. The research methodology has introduced us to a time-convenient approach for investigating polarization conversion metasurfaces.

A deep learning model will be introduced to provide on-demand polarization conversion. We discussed earlier that a pattern-producing network(CPPN) will be built with the help of generative adversarial networks(GANs). The CPPN will generate hundreds of compositional patterns based on the desired functionalities. We introduced the forward prediction simulator model in this work to validate those patterns. Numerical verification will be performed on those validated metasurfaces to compare the performance. Finally, an experimental verification will be carried out to confirm the research objective.

Bibliography

[1] D. Dai, L. Liu, S. Gao, D.-X. Xu, and S. He, "Polarization management for silicon photonic integrated circuits," *Laser & Photonics Reviews*, vol. 7, no. 3, pp. 303–328, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/lpor.201200023

[2] X. Fang, K. F. MacDonald, E. Plum, and N. I. Zheludev, "Coherent control of light-matter interactions in polarization standing waves," *Scientific reports*, vol. 6, no. 1, p. 31141, 2016.

[3] J. E. Solomon, "Polarization imaging," *Appl. Opt.*, vol. 20, no. 9, pp. 1537–1544, May 1981. [Online]. Available: https://opg.optica.org/ao/abstract.cfm?URI=ao-20-9-1537

[4] V. Karassiov, "Polarization of light in classical and quantum optics: Concepts and applications," *Optics and Spectroscopy*, vol. 103, pp. 137–144, 2007.

[5] Y. P. Svirko and N. I. Zheludev, *Polarization of light in nonlinear optics*, 2000.

[6] C. Guo, F. Liu, S. Chen, C. Feng, and Z. Zeng, "Advances on exploiting polarization in wireless communications: Channels, technologies, and applications," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 1, pp. 125–166, 2017.

[7] V. Thilak, D. G. Voelz, and C. D. Creusere, "Polarization-based index of refraction and reflection angle estimation for remote sensing applications," *Appl. Opt.*, vol. 46, no. 30, pp. 7527–7536, Oct 2007. [Online]. Available: https://opg.optica.org/ao/abstract.cfm?URI=ao-46-30-7527

[8] A. H. Dorrah, N. A. Rubin, A. Zaidi, M. Tamagnone, and F. Capasso, "Metasurface optics for on-demand polarization transformations along the optical path," *Nature Photonics*, vol. 15, no. 4, pp. 287–296, 2021.

[9] X.-J. Shang, X. Zhai, J. Yue, X. Luo, J.-P. Liu, X.-P. Zhu, H.-G. Duan, and L.-L. Wang, "Broad-band and high-efficiency polarization converters around 1550 nm based on composite structures," *Opt. Express*, vol. 25, no. 13, pp. 14 406–14 413, Jun 2017. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?URI=oe-25-13-14406

[10] Y. Sun, Y. Liu, T. Wu, Y. Wang, J. Li, H. Ye, H. Fan, and X. Wang, "All-dielectric metasurface for linear-polarization conversion with an arbitrary polarization rotating

angle," *Optics and Laser Technology*, vol. 157, p. 108762, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030399222009082

[11] Z. Liu, D. Zhu, S. P. Rodrigues, K.-T. Lee, and W. Cai, "Generative model for the inverse design of metasurfaces," *Nano letters*, vol. 18, no. 10, pp. 6570–6576, 2018.

[12] D. Liu, Y. Tan, E. Khoram, and Z. Yu, "Training deep neural networks for the inverse design of nanophotonic structures," *Acs Photonics*, vol. 5, no. 4, pp. 1365–1369, 2018.

[13] Z. Liu, D. Zhu, K.-T. Lee, A. S. Kim, L. Raju, and W. Cai, "Compounding meta-atoms into metamolecules with hybrid artificial intelligence techniques," *Advanced Materials*, vol. 32, no. 6, p. 1904790, 2020.

[14] https://www.edmundoptics.com/knowledge-center/application-notes/optics/introduction-to-polarization/, [Accessed 22-08-2024].

[15] "Circular polarization - Wikipedia — en.wikipedia.org," https://en.wikipedia.org/wiki/Circular_polarization, [Accessed 22-08-2024].

[16] A. Sihvola, "Metamaterials in electromagnetics," *Metamaterials*, vol. 1, no. 1, pp. 2–11, 2007.

[17] W. J. Padilla, D. N. Basov, and D. R. Smith, "Negative refractive index metamaterials," *Materials today*, vol. 9, no. 7-8, pp. 28–35, 2006.

[18] V. G. Veselago, "The electrodynamics of substances with simultaneously negative values of ε and μ," *Soviet Physics Uspekhi*, vol. 10, no. 4, p. 509, apr 1968. [Online]. Available: https://dx.doi.org/10.1070/PU1968v010n04ABEH003699

[19] D. R. Smith, J. B. Pendry, and M. C. Wiltshire, "Metamaterials and negative refractive index," *science*, vol. 305, no. 5685, pp. 788–792, 2004.

[20] J. B. Pendry, "Negative refraction makes a perfect lens," *Physical review letters*, vol. 85, no. 18, p. 3966, 2000.

[21] B. Zhang, Y. Luo, X. Liu, and G. Barbastathis, "Macroscopic invisibility cloak for visible light," *Physical Review Letters*, vol. 106, no. 3, p. 033901, 2011.

[22] Y. Dong and T. Itoh, "Metamaterial-based antennas," *Proceedings of the IEEE*, vol. 100, no. 7, pp. 2271–2285, 2012.

[23] A. Ebrahimi, W. Withayachumnankul, S. Al-Sarawi, and D. Abbott, "High-sensitivity metamaterial-inspired sensor for microfluidic dielectric characterization," *IEEE Sensors Journal*, vol. 14, no. 5, pp. 1345–1351, 2013.

[24] S. Larouche, Y.-J. Tsai, T. Tyler, N. M. Jokerst, and D. R. Smith, "Infrared metamaterial phase holograms," *Nature materials*, vol. 11, no. 5, pp. 450–454, 2012.

[25] N. I. Landy, S. Sajuyigbe, J. J. Mock, D. R. Smith, and W. J. Padilla, "Perfect metamaterial absorber," *Physical review letters*, vol. 100, no. 20, p. 207402, 2008.

[26] J. Hu, S. Bandyopadhyay, Y.-h. Liu, and L.-y. Shao, "A review on metasurface: from principle to smart metadevices," *Frontiers in Physics*, vol. 8, p. 586087, 2021.

[27] H.-T. Chen, A. J. Taylor, and N. Yu, "A review of metasurfaces: physics and applications," *Reports on progress in physics*, vol. 79, no. 7, p. 076401, 2016.

[28] N. Yu and F. Capasso, "Flat optics with designer metasurfaces," *Nature materials*, vol. 13, no. 2, pp. 139–150, 2014.

[29] Z. Wei, Y. Cao, X. Su, Z. Gong, Y. Long, and H. Li, "Highly efficient beam steering with a transparent metasurface," *Optics express*, vol. 21, no. 9, pp. 10 739–10 745, 2013.

[30] L. Huang, S. Zhang, and T. Zentgraf, "Metasurface holography: from fundamentals to applications," *Nanophotonics*, vol. 7, no. 6, pp. 1169–1190, 2018.

[31] J. Balthasar Mueller, N. A. Rubin, R. C. Devlin, B. Groever, and F. Capasso, "Metasurface polarization optics: independent phase control of arbitrary orthogonal states of polarization," *Physical review letters*, vol. 118, no. 11, p. 113901, 2017.

[32] M. Jang, Y. Horie, A. Shibukawa, J. Brake, Y. Liu, S. M. Kamali, A. Arbabi, H. Ruan, A. Faraon, and C. Yang, "Wavefront shaping with disorder-engineered metasurfaces," *Nature photonics*, vol. 12, no. 2, pp. 84–90, 2018.

[33] S. Zhang, C. L. Wong, S. Zeng, R. Bi, K. Tai, K. Dholakia, and M. Olivo, "Metasurfaces for biomedical applications: imaging and sensing from a nanophotonics perspective," *Nanophotonics*, vol. 10, no. 1, pp. 259–293, 2020.

[34] W. Ma, Z. Liu, Z. A. Kudyshev, A. Boltasseva, W. Cai, and Y. Liu, "Deep learning for the design of photonic structures," *Nature Photonics*, vol. 15, no. 2, pp. 77–90, 2021.

[35] P. R. Wiecha, A. Arbouet, C. Girard, and O. L. Muskens, "Deep learning in nanophotonics: inverse design and beyond," *Photonics Research*, vol. 9, no. 5, pp. B182–B200, 2021.

[36] T. Qiu, X. Shi, J. Wang, Y. Li, S. Qu, Q. Cheng, T. Cui, and S. Sui, "Deep learning: a rapid and efficient route to automatic metasurface design," *Advanced Science*, vol. 6, no. 12, p. 1900128, 2019.

[37] Y. Ma and Y. Hao, "Deep learning in metasurface design and optimization," in *Metamaterials-by-Design*. Elsevier, 2024, pp. 203–232.

[38] S. K. Melanthota, D. Gopal, S. Chakrabarti, A. A. Kashyap, R. Radhakrishnan, and N. Mazumder, "Deep learning-based image processing in optical microscopy," *Biophysical Reviews*, vol. 14, no. 2, pp. 463–481, 2022.

[39] B. Karanov, M. Chagnon, V. Aref, F. Ferreira, D. Lavery, P. Bayvel, and L. Schmalen, "Experimental investigation of deep learning for digital signal processing in short reach optical fiber communications," in *2020 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2020, pp. 1–6.

[40] G. Fumero, G. Batignani, E. Cassetta, C. Ferrante, S. Giagu, and T. Scopigno, "Retrieving genuine nonlinear raman responses in ultrafast spectroscopy via deep learning," *APL Photonics*, vol. 9, no. 6, 2024.

[41] A. Merchant, S. Batzner, S. S. Schoenholz, M. Aykol, G. Cheon, and E. D. Cubuk, "Scaling deep learning for materials discovery," *Nature*, vol. 624, no. 7990, pp. 80–85, 2023.

[42] J. M. Benítez, J. L. Castro, and I. Requena, "Are artificial neural networks black boxes?" *IEEE Transactions on neural networks*, vol. 8, no. 5, pp. 1156–1164, 1997.

[43] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic programming and evolvable machines*, vol. 8, pp. 131–162, 2007.

[44] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[45] S. An, B. Zheng, M. Y. Shalaginov, H. Tang, H. Li, L. Zhou, J. Ding, A. M. Agarwal, C. Rivero-Baleine, M. Kang *et al.*, "Deep learning modeling approach for metasurfaces with high degrees of freedom," *Optics Express*, vol. 28, no. 21, pp. 31 932–31 942, 2020.

[46] N. B. Roberts and M. Keshavarz Hedayati, "A deep learning approach to the forward prediction and inverse design of plasmonic metasurface structural color," *Applied Physics Letters*, vol. 119, no. 6, 2021.

[47] C. C. Nadell, B. Huang, J. M. Malof, and W. J. Padilla, “Deep learning for accelerated all-dielectric metasurface design,” *Optics express*, vol. 27, no. 20, pp. 27 523–27 535, 2019.

[48] Y. Teng, C. Li, S. Li, Y. Xiao, and L. Jiang, “Efficient design method for terahertz broadband metasurface patterns via deep learning,” *Optics & Laser Technology*, vol. 160, p. 109058, 2023.

[49] M. Hussain, J. J. Bird, and D. R. Faria, “A study on cnn transfer learning for image classification,” in *Advances in Computational Intelligence Systems: Contributions Presented at the 18th UK Workshop on Computational Intelligence, September 5-7, 2018, Nottingham, UK*. Springer, 2019, pp. 191–202.

[50] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of cnn and rnn for natural language processing,” *arXiv preprint arXiv:1702.01923*, 2017.

[51] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, “Medical image classification with convolutional neural network,” in *2014 13th international conference on control automation robotics & vision (ICARCV)*. IEEE, 2014, pp. 844–848.

[52] Y. Fan, X. Lu, D. Li, and Y. Liu, “Video-based emotion recognition using cnn-rnn and c3d hybrid networks,” in *Proceedings of the 18th ACM international conference on multimodal interaction*, 2016, pp. 445–450.

[53] K. O’shea and R. Nash, “An introduction to convolutional neural networks,” *arXiv preprint arXiv:1511.08458*, 2015.

[54] S. Sharma, S. Sharma, and A. Athaiya, “Activation functions in neural networks,” *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[55] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, “Understanding batch normalization,” *Advances in neural information processing systems*, vol. 31, 2018.

[56] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, “On empirical comparisons of optimizers for deep learning,” *arXiv preprint arXiv:1910.05446*, 2019.

[57] S. Targ, D. Almeida, and K. Lyman, “Resnet in resnet: Generalizing residual architectures,” *arXiv preprint arXiv:1603.08029*, 2016.

[58] J. Liang, “Image classification based on resnet,” in *Journal of Physics: Conference Series*, vol. 1634, no. 1. IOP Publishing, 2020, p. 012110.

[59] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.

[60] E. D. Palik, *Handbook of optical constants of solids*. Academic press, 1998, vol. 3.

[61] A. Taflove, A. Oskooi, and S. G. Johnson, *Advances in FDTD computational electrodynamics: photonics and nanotechnology*. Artech house, 2013.

[62] S. D. Gedney, *Introduction to the finite-difference time-domain (FDTD) method for electromagnetics*. Morgan & Claypool Publishers, 2011, vol. 27.

[63] “Finite-difference time-domain method - Wikipedia — en.wikipedia.org,” https://en.wikipedia.org/wiki/Finite-difference_time-domain_method, [Accessed 23-08-2024].

[64] C. Wang, H.-X. Xu, Y. Wang, G. Hu, H. Luo, and K. Wang, “Reconfigurable transmissive metasurface synergizing dynamic and geometric phase for versatile polarization and wavefront manipulations,” *Materials and Design*, vol. 225, p. 111445, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0264127522010681

[65] X. Zhu, Y. Cheng, J. Fan, F. Chen, H. Luo, and L. Wu, “Switchable efficiency terahertz anomalous refraction and focusing based on graphene metasurface,” *Diamond and Related Materials*, vol. 121, p. 108743, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925963521005069

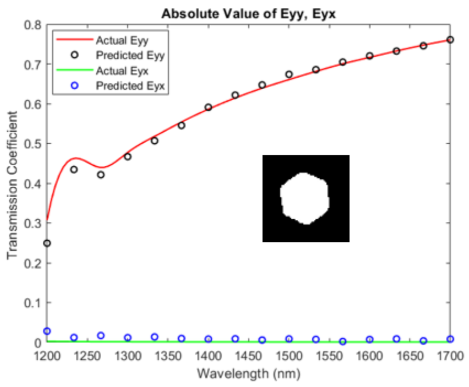
[66] M.-E. Mustafa, M. Amin, O. Siddiqui, and F. A. Tahir, “Quasi-crystal metasurface for simultaneous half-and quarter-wave plate operation,” *Scientific reports*, vol. 8, no. 1, p. 15743, 2018.

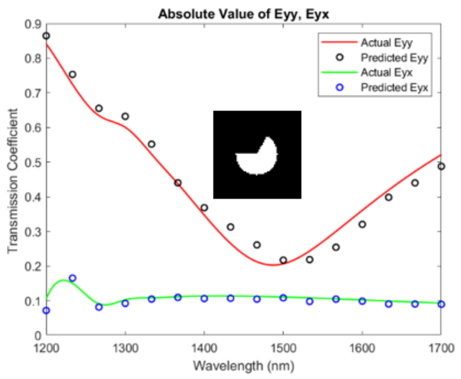
[67] J. Parmar, S. K. Patel, and V. Katkar, “Graphene-based metasurface solar absorber design with absorption prediction using machine learning,” *Scientific Reports*, vol. 12, no. 1, p. 2609, 2022.

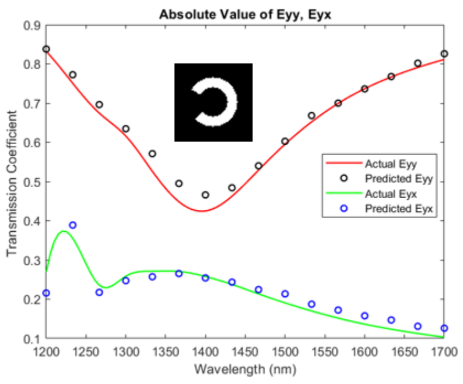
A Appendix A

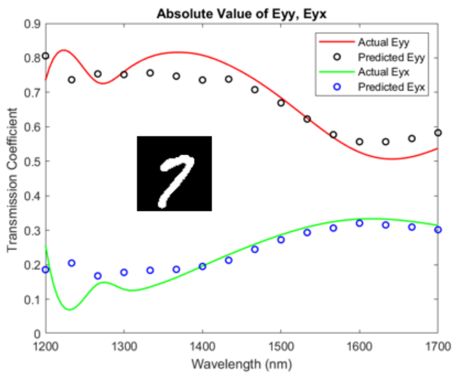
All the necessary codes and the neural network model can be found at the following link: GitHub

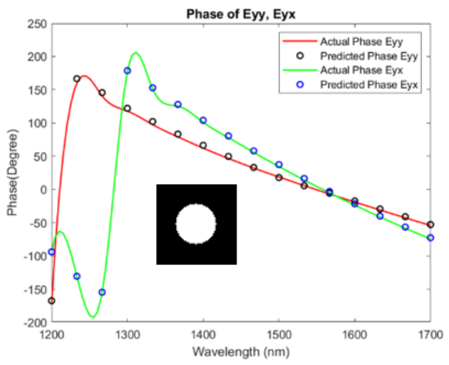
A.1 Evaluation of The Test Datasets for Y-polarized Light

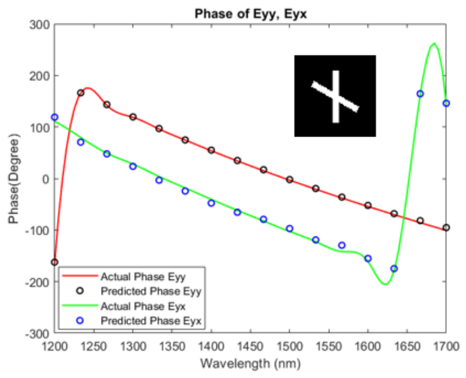






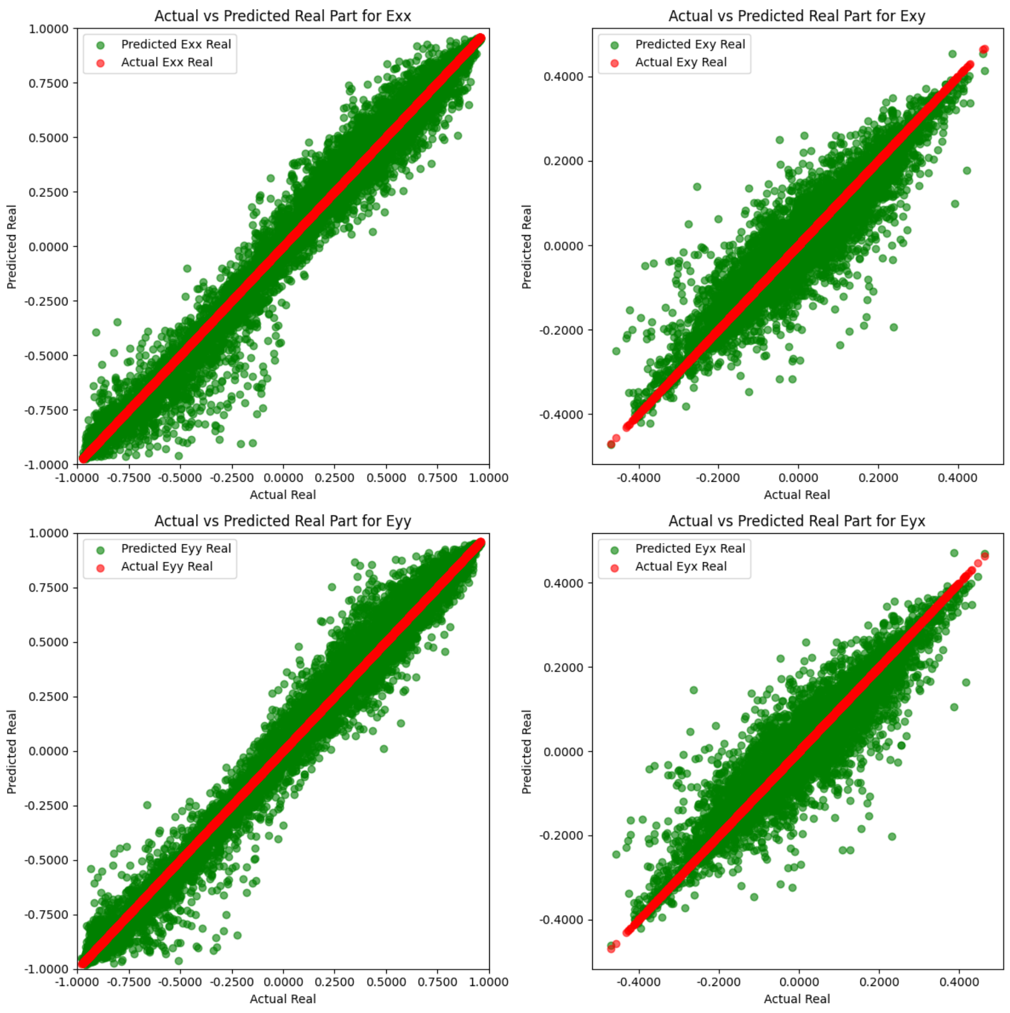




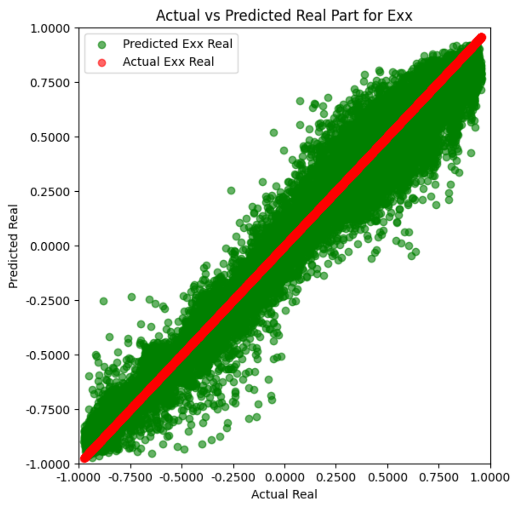


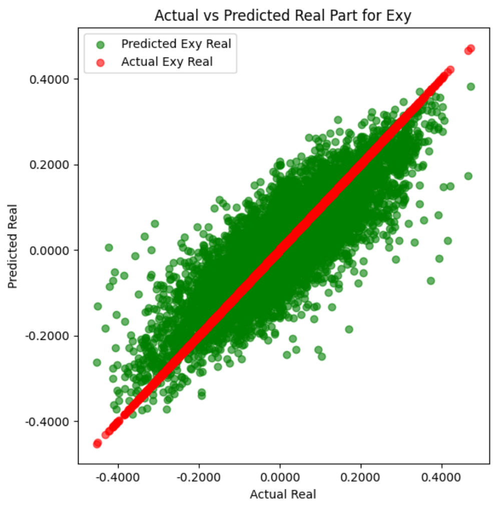
A.2 Accuracy Statistics for Different Optimizer

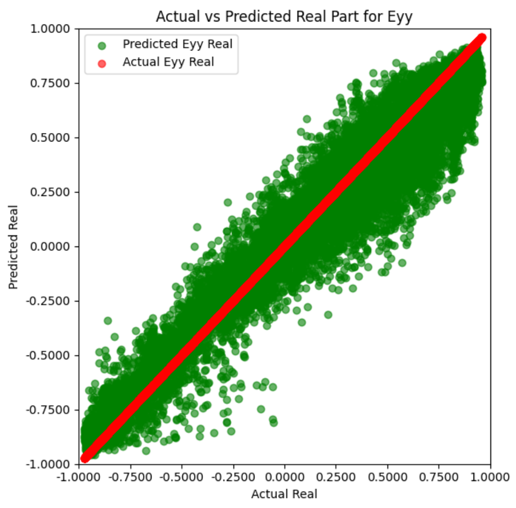
A.2.1 Accuracy Statistics for Adam Optimizer

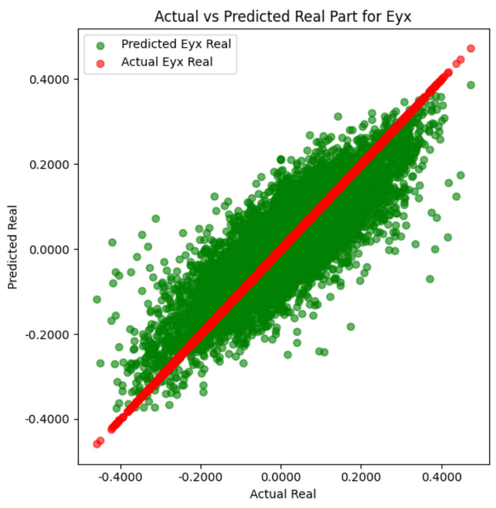


A.2.2 Accuracy Statistics for RMSprop Optimizer

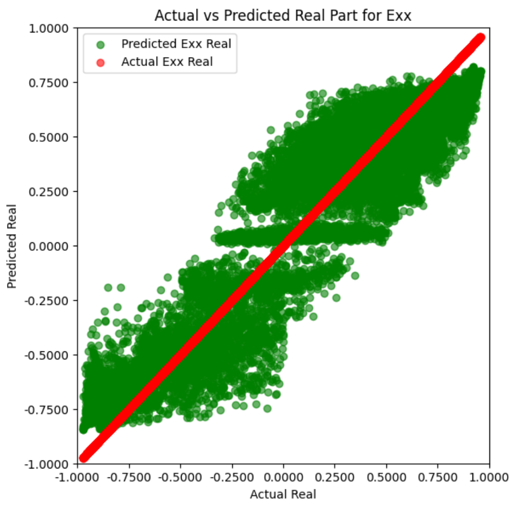


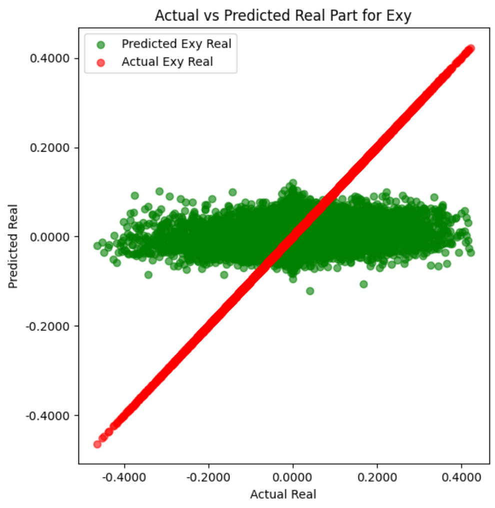


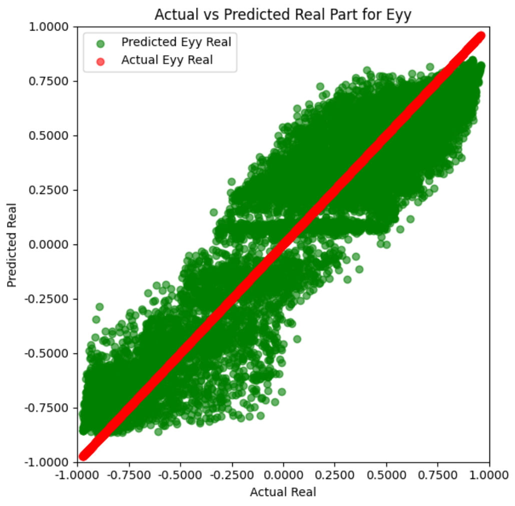


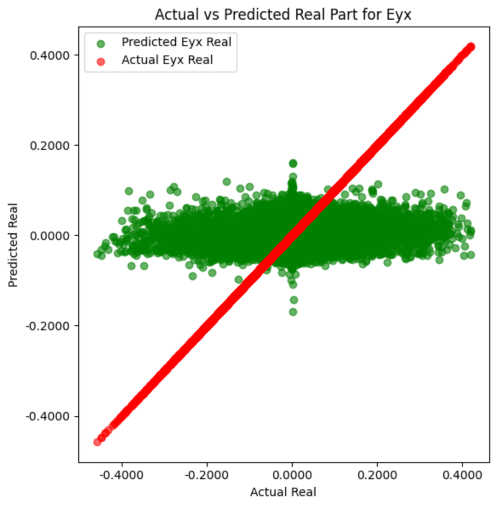


A.2.3 Accuracy Statistics for SGD Optimizer

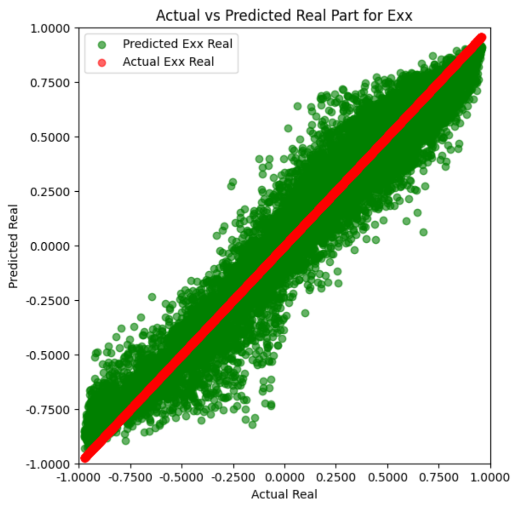


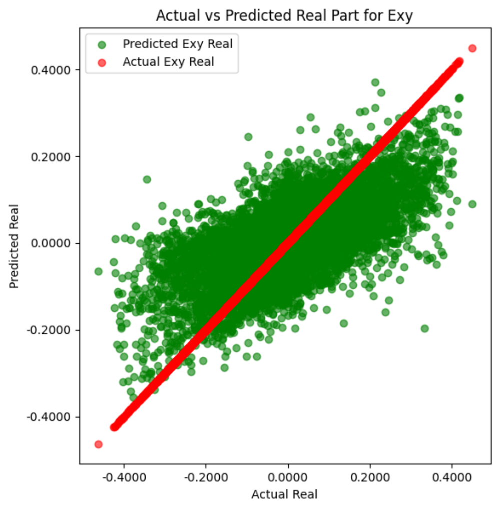


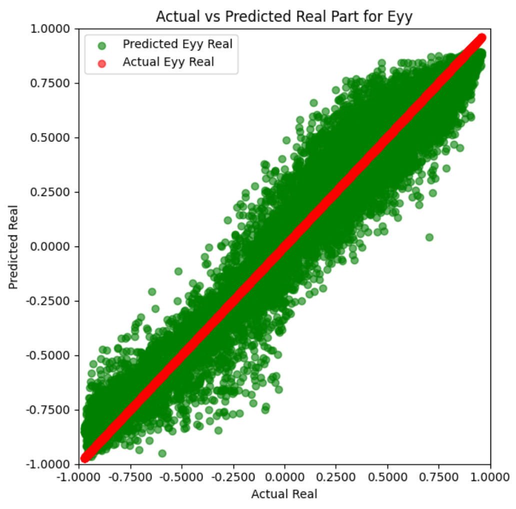


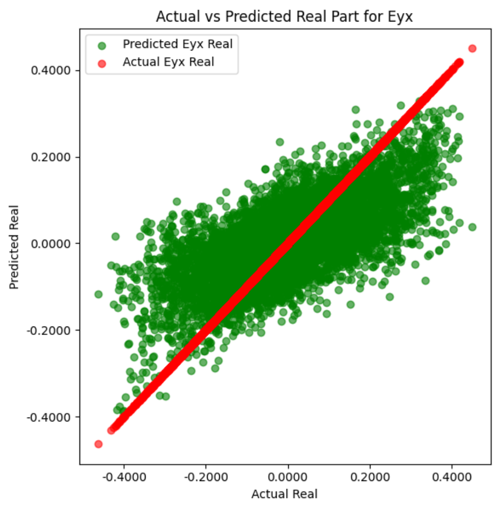


A.2.4 Accuracy Statistics for Adagrad Optimizer









Statement of Non-plagiarism

I hereby declare that all information in this report has been obtained and presented in accordance with academic rules and ethical conduct and the work I am submitting in this report, except where I have indicated, is my own work.

Supervisor Approval

I, the undersigned, Prof. Humeyra Caglayan, supervisor of Md Imran Hossain, a student of the PSRS EMJMD, during his master thesis at Tampere University certify that I approve the content of this master thesis report entitled “A Deep Learning Approach for Predicting Transmission Spectra of Metasurfaces”.

[illegible]

25.08.2024